Trip-A-Tron

MANUAL

LLAMASOFT

© 1988

CONTENTS

Trip·A·TRON

- AN INTRODUCTION

1: What is TRIP-A-TRON?

Trip-A-Tron is a LIGHT SYNTHESISER. The lightsynth is a concept which I have been working on for the last few years, using a wide variety of computers. The earliest form of lightsynth was programmed on a Commodore 64, and was called PSYCHEDELIA. The program has evolved through COLOURSPACE on the 8-bit and then 16-bit Atari machines, and has now been completely re-written and made ludicrously powerful to become TRIP-A-TRON.

2: Yeah, Great, but what does it DO?

The lightsynth is a new kind of graphics program. Up till now, graphics programs have used the computer to produce results which are directly analogous to ‘real-world’ counterparts; for example, a paint program provides the user with tools to produce the definite result of ‘a picture’, and an animation program is used to make a ‘film’ or ‘cartoon’. Pictures, films and animation can all be done without computers; the machine just makes the work easier.

In creating the lightsynth, I sought to create a new artform entirely unique to the computer. A lightsynth show has no direct counterpart outside the machine. (Perhaps laser-shows and rock-concert lightshows come close, but they are largely computer-generated anyway). A lightsynth performance is continuously changing; the images are seldom static, and usually do not attempt to communicate any literal meaning. The display is the visual equivalent of a piece of music; both have dimension in time, both consist of a succession of complex rhythms and harmonies – and both can be performed, in realtime, by a skilled human performer.

3: So can I plug my Stereo Into It Then?

No.

3a: Why not?

Because Trip-a-Tron is a synthesiser, not a sound-to-light program. It requires that a human operator provide dynamic input in order to make it work. The reason for this is simple: no matter how complex the hardware, no matter how clever the software, no automatic device will ever be able to visually interpret music in the same way as a human being. Music is not, after all, just a bunch of rhythms and harmonies; it consists largely of something no machine can detect – emotion. To fully express a piece of music graphically inevitably needs the human touch. To make an ‘automatic’ lightsynth would be as pointless as trying to make a robot guitar which played itself. You might make it sound OK in a mathematical sort of way, but it’d never be able to play the guitar solo from the end of ‘Comfortably Numb’.

4: Why a light synthesiser? Why not a light-ukelele?

Well, the function of the lightsynth parallels closely many of the operations used by sound-synthesisers. The main difference, of course, is that where a soundsynth creates and modifies sounds, the lightsynth creates and modifies colour, symmetry and motion. Both types of synthesiser make use of waveforms, oscillators, keyboards, sequencers and various input devices. Both are capable of producing almost infinitely varied output. In the same way that a synth player can ‘build’ his own sounds, the lightsynth operator can invent and perform graphic effects, continuously, in real-time.

5: Wow. Sounds heavy. Do I need a degree in Maths?

Of course not! The lightsynth has been designed in such a way that you can choose to go into as little – or as much – depth as you desire. Rewarding results may be obtained with even a small knowledge of the basic controls, and as you grow more confident, you can begin to explore and exploit the power of the more complex functions.

Perhaps this is the right time to point out one of the main differences between lightsynth and conventional, musical instruments – lightsynth is considerably more pleasant to learn! When you’re getting started on a musical instrument, there is often a long frustrating ‘complete beginner’ type phase during which your best efforts can sound awful. On the lightsynth, it is almost impossible to play the visual equivalent of a ‘bum note’ – just by the nature of the symmetry inherant in the generated displays, even quite random scribblings still come out looking groovy. All that happens is that as your skill increases your music displays become more ‘appropriate’, ‘in-sync’ with the particular music you perform to. Music and graphics naturally go well together, but a well-performed lightsynth accompaniement combines synergistically with the music to become something entirely different and very powerful. You’ll know it when you get there.

6: What Kind of Music Goes Best?

Whatever your favourite music is, it’ll go well with Trip-A-Tron. The program is capable of a range of effects suitable for any kind of music – fast, slow, heavy, classical, whatever. The only requirement is that you, as the performer, are able to groove on the music you’re playing to.

7: OK then, blow my mind

Right. Proceed to the next section with disks in hand.

GETTING STARTED WITH TRIP-A-TRON

You will need:
Your program disks
An Atari ST computer, and a colour monitor or TV
This manual, open on your knee
A cup of tea
A little patience

Okay. First thing, put some music on, anything, it doesn't matter, we're not getting serious yet, some Floyd'd be nice. Just be relaxed. Don't turn the lights off just yet, you need to see what you're doing. Turn on your computer and grab a couple of spare disks. First thing is to MAKE A BACKUP COPY of the Trip-A-Tron disks. The program is not protected, so you can copy it onto 'working' disks and keep your masters safe in case of disk failure. Follow the standard Atari ST method for copying disks and make a copy of each of the program disks.

Note for double-sided and hard disk drive users

If you have a double-sided drive, you can make life easier for yourself by putting all the program files on one disk, eliminating the need for any disk swapping during the load. Format a DS disk, and copy all the files from the Trip-A-Tron 'A' disk onto it. Then, copy the TRIP.DAT file from Trip-A-Tron 'B' onto the same disk. Provided you boot from drive 'A', all will be cool. Hard disk users can copy the program onto hard drive with no problems – just ensure that TRIP.DAT and TRIP.PRG are in the same directory (I.e don't have one inside a folder and the other outside!). I would beg of you hard drivers to just be cool for a moment and boot off the floppy the first time – or at least leave either 'A' or 'B' Trip-A-Tron disk in drive 'A' – because the system demo files expect data on drive 'A'. I'll fill you in on how to make the demos boot off drives other than 'A' a bit later.

So now you've got your backups. Now:

REMOVE your desk accessories if you have any (especially important for 520 owners; DA's eat memory!). Reboot if necessary with a TRIP-A-TRON disk in the drive.

IF you have a SINGLE-DRIVE SINGLE-SIDED system:

Place the Trip-A-Tron 'A' disk in the drive. Double-click the disk to open the directory window. Find TRIP.PRG and double click to load the program. Halfway through loading you'll see an alert box telling you to insert disk B; swap the disks and press RETURN. The program will finish loading and announce itself. Press the LEFT mouse button to clear the title display. You'll be asked to replace the 'A' disk. Do so if you like, but the demo files are recorded on both disks so you don't have to: just press RETURN. The drive will go as the demo files are macro_loaded, and then the Control Panel will rise up, and you're ready to go.

IF you have a DUAL-DRIVE system:

Place Trip-A-Tron 'A' disk in drive A, and the 'B' disk in drive B. Double-click the 'A' disk icon to open the directory window. Find TRIP.PRG and double-click to load the program. When it's loaded you'll see the TRIP-A-TRON title. Press the LEFT Mouse button to clear the display. There'll be a little disk activity on A as the demo files are macro_loaded, and then the Panel will rise and you're up and running.

IF you have TRIP-A-TRON on a DOUBLE-SIDED disk

Boot from drive A and follow instructions as for a dual-drive system.

The System Demos – First Encounter

You're loaded, the Panel is up, no bombs, no disk errors, good. Have a swig of that tea. Don't worry about the control panel for now, we'll come to that in detail a bit later on. Right now, just press ESC to put the panel away and leave you with a nice clear screen.

You'll notice a little dot on the screen. Move your mouse and the dot follows. That little dot is your cursor; you use the mouse to move the dot and orchestrate your display. Get used to moving that dot. When you're quite happy with moving the dot around, try moving the dot while you hold down the left mouse button.

Yow! Graphics! Yeah! Play around with them for a little while. Notice how slow and fast mouse motions produce different results. Try fine subtle mouse motions, great sweeping ones, whatever; just play with the graphics and get used to the feel. Now, try pressing any of the top row of keys from figure '1' to Backspace (don't press ESC, that calls back the Control Panel. If you do that by accident, don't worry, just press ESC again to hide the panel once more). Pressing these keys will randomly choose a new palette for you. Notice how the dynamics of colour flow enhance the patterns. Play with the palette-change for a while.

Okay, now onto the fundamental core of lightsynth display: symmetry. Try pressing any keys in the cursor cluster; Help, Undo, Insert, Clr/Home, and the arrow keys. These keys change the symmetry settings. You'll have a lot of fun with just those eight keys. You'll see how, as more symmetry is added, the inertia of pattern flow becomes 'heavier'. As you learn you'll become used to this and modify your mouse motions accordingly.

Now that you've got used to the feel for the way it works, just experiment. Use any of the keys on the main keyboard (except ESC) to see some of the effects you can generate using Trip-a-tron. Just so you have an idea what's causing what on the keyboard, keys Q-P produce decay-mode effects, keys A-L offer line-mode patterns and the row of keys between the two SHIFT keys produce 'expandor' effects. (More on this terminology later). Remember that small delicate mouse motions can produce excellent results, and try to be smooth in your mouse movements. If the system ever seems to 'clog up' with too much symmetry etc., pressing the SPACE bar will return you to a less hectic mode.

Pattern-generation such as that displayed in the demo is only the beginning of what the program can do. When you're through playing with the pattern demo, press function key F10. This will load another demo file, with completely different effects laid out on the keyboard: Laser and Starfield effects. Once the new demo has loaded, try pressing any of the bottom row of keys to see some starfield effects. With many of the starfields, it's possible to control the display by moving the mouse (you don't have to press any mouse buttons but sometimes the right-hand button does something interesting)… The '?' key will turn off all star effects, and if you really get into trouble just press SPACE to turn everything off. Starfield symmetries can be changed just like the pattern symmetries were: use the cursor cluster.

The upper row of keys on the main keyboard activate the two built-in 'laser' effects channels. Laser-effects are mathematically-generated oscilloscope effects drawn either as a series of dots or lines. The generated shapes can be merged with those of the starfield, and their symmetries can be altered in a similar manner. Some of them respond to mouse movements. To shutdown the Laser Channels, press the underline (-) key to turn off channel one, and Backspace to turn off channel two.

There are a bunch of palettes laid out on the numeric-pad keys, as well as a few strobes. Experiment and have fun. Pressing the function key F10 will re-load the original pattern-generation demo from disk, so you can sit and play with the demos easily without ever needing the Control Panel.

Here is a key layout summary for the demo programs:

Demo 1 (Pattern Generation – use mouse and LEFT hand button)

1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, =, ', BACKSPACE
- Change randomly to a new palette setting

Q, W, E, R, T, Y, U, I, O, P, [,]
- Various Decay-Mode effects

A, S, D, F, G, H, J, K, L
- Various Line-Mode effects

Key ;
- Turn ON Secondary Cursor

Key '
- Turn OFF Secondary Cursor

\, Z, X, C, V, B, N, M, <, >, /
Various Expander and Mixed-Mode Effects

SPACE
- Panic button

HELP, UNDO, INSERT, CLR/HOME, arrow-keys
- Change pattern symmetries

Numeric-pad
- A selection of strobe effects

When you've read some more of this manual, here's some more to try on Demo 1:

DELETE: Prepare to change pre-symmetry (next symmetry command affects PreSym change)
F1: Toggle presym on/off
F2: Toggle REMAP on/off (needs .MAP file loaded)
F3: Toggle CUBIC MAP on/off
F4: Toggle SHEAR on/off
F5: Toggle RESIZING on/off
F6: Toggle postsym (main symmetry) on/off
F10: Chain DEMO 2 from drive A

Demo 2 (Starfield, Laser demo)

1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, =
- Turn on various LASER effects

Key '
- Shut down Laser 1

BACKSPACE
- Shut down Laser 2

A, S, D, F, G, H, J, K, L
- Starfield effects

Key ?
- Shut down starfield generation

HELP, UNDO, INSERT, CLR/HOME, arrow keys
- Change main symmetry (postsym)

Numeric Keypad
- Various palettes and STROBE fx

Delete
- Get ready for Presym setup (next Symmetry command affects Presym). This is useful in this demo, becase the laser effects which consist of lines pick up their symmetry from the Presym setting.

Space Bar
- Panic button

F10
- Chain DEMO 1

Okay, I'll assume that you've had a good go of the demo setups, and are ready to learn some more about the lightsynth. Switch off starfield and laser generation if you've got any going, then press ESC to bring up the control panel.

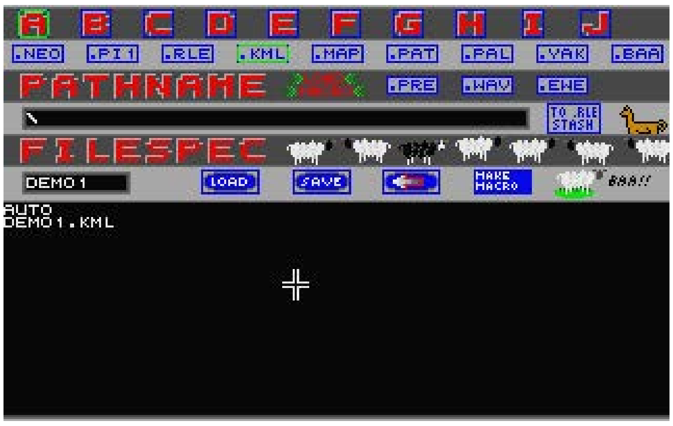
Here's where you'll start to really learn about what the lightsynth can do. That control panel provides you with access to systems which allow you complete control over all the lightsynth's functions. Before we plough on in, study the Control Panel and we'll take a look at the various modules, explaining what they do.

TRIP-A-TRON SYSTEM CONTROL: OVERVIEW AND NAMING OF PARTS

Take a look at the Control Panel. We'll name each icon in turn, explaining briefly what it is and where its function lies in the Trip-A-Tron system. As we go, I'll give you instructions on how to get into specific control screens associated with each icon. Until you've read some more about each specific function don't bother trying to change anything; just get used to entering and leaving the various control screens.

Fine. Starting at the top-left of the main icon panel (ignore the Colour Presets for now).

DISK ACCESS



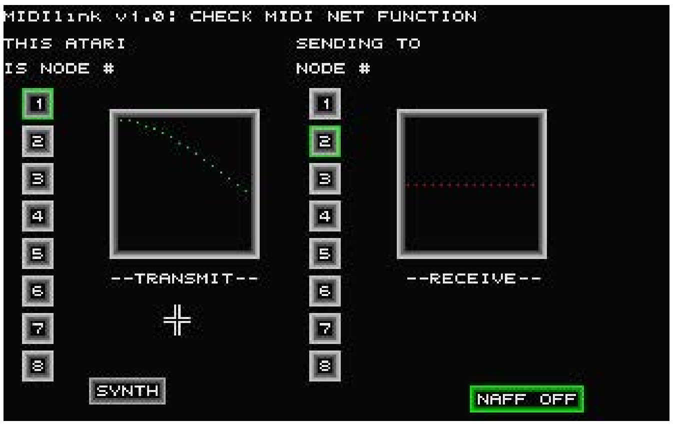
```
AUTO
DEMO1.KML
```

To enter the Disk screen: press either Mouse button on the Disk Access icon.

To leave the Disk screen: click on the little sheep with 'BAA' written underneath it (halfway down the screen, right-hand side).

The Disk Access screen is used for SAVEing and LOADing the various data files used by the Lightsynth. This is where you can go to load programs, patterns, palettes and suchlike; you can also make 'macro_load' files from this screen (macro_loaded files are just a list of all the different files and sequencs used in a display. A whole set of such files can be made to load automatically).

MIDI SETUP AND TEST



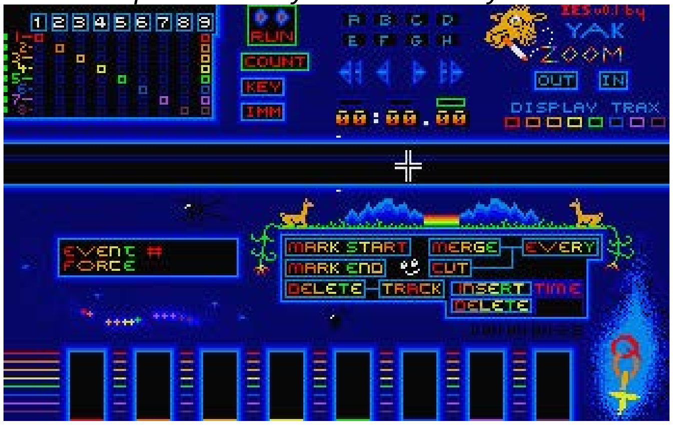
To enter MIDI screen: Click either Mouse button on MIDI icon

To leave MIDI screen: Click on 'Naff Off' (bottom-left of screen)

From Midi Test screen you can check out the function of any nodes you're communicating with on the Midi network, set your own local node number, or set Synth mode (for 'listening' to a remote MIDI device).

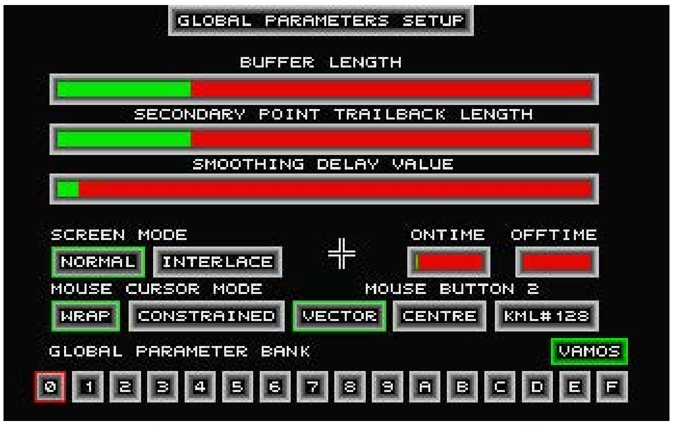
INTERNAL EVENT SEQUENCER

NOTE: Sequencer is only accessible to systems with 1 Meg memory or more.



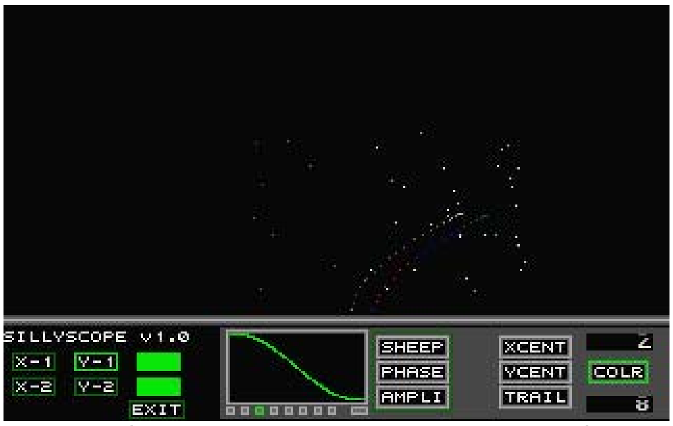
To enter IES: Click the RIGHT Mouse Button over the Sequencer icon. If this is the first time you've used the IES this session, an alert box will warn you that the sequencer's going to steal a frame of video-sequencer memory. Click on the GO FUR IT to tell the system to go ahead, and after a brief pause the IES screen will be displayed.
To leave IES: Click on the large colourful QUIT sign (bottom right of screen).
The Internal Event Sequencer is like a simple inbuilt 8-track recorder for information such as strobes, symmetry changes, animation sequences – in fact anything you can write a KML program for. Basically, you can enter rhythms and sync points from the keyboard, assign each 'note' in the sequence a different KML program-number; when you start the IES at performance-time, the symmetry changes and strobes take place automatically at the correct moment.

GLOBAL PARAMETERS



To enter Global Parameters screen: Press either mouse button over the Global Parameters icon.
To leave Global Parameters screen: Click on VAMOS (bottom right of screen).
The Global Parameters screen is used to adjust commonly used pattern parameters and mouse control options. Up to sixteen sets of Global Parameter definitions can be held at once, and switched between using either this screen or KML

THE SILLYSCOPE

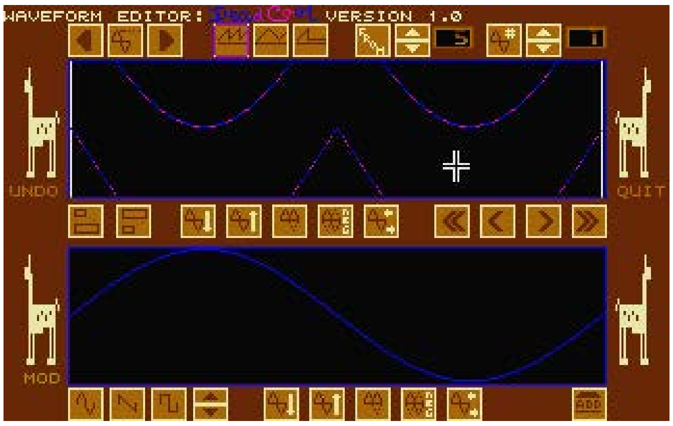


To enter Sillyscope screen: press RIGHT mouse button over Sillyscope icon.
To leave Sillyscope screen: click on EXIT (Bottom left, in a bit)
The sillyscope screen controls the two Laser Generators. You use this screen as a ‘workshop’ to build the effects you want, then put the settings discovered into your KML programs to set up the Laser Generators.

LLAMA ICON
This is just an ‘About TRIP-A-TRON’ type icon. Click on it to see the current version number. An alert box with the information will appear. Just click in the alert boxes’ Quit Box to leave.

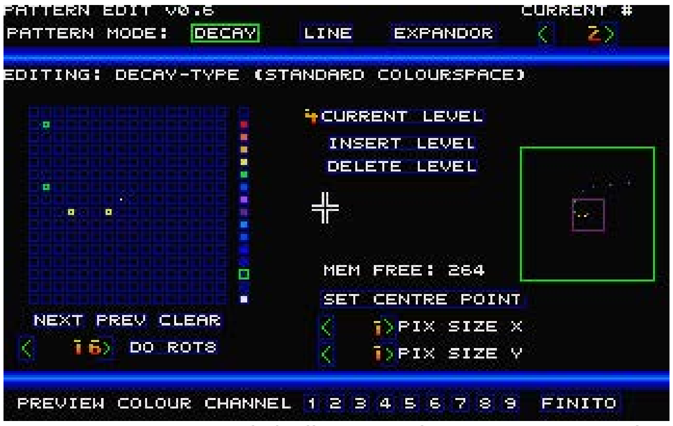
Middle row, starting at leftmost icon

WAVEFORM BUILD/CONTROL



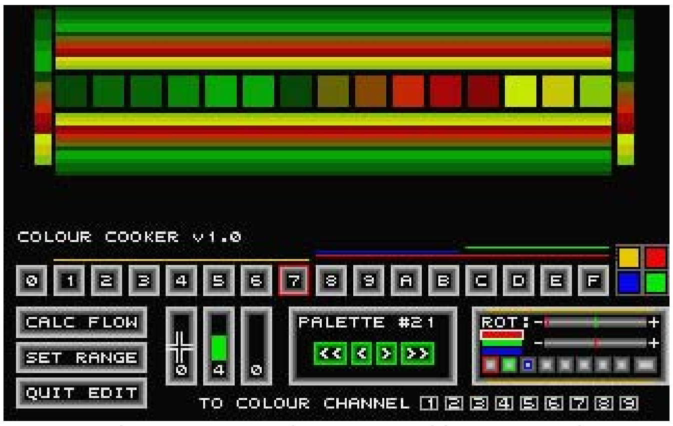
This screen has two functions: Build Waveforms and Oscillator FX
To enter Build Waveforms: press the RIGHT mouse button over the Waveform icon.
To leave Build Waveforms: click on the Quit-Llama (upper right of screen)
The Build Waveforms screen is used to construct up to eight different waveforms. These waveforms can be used in a variety of ways, for things like RGB colour oscillators, Sillyscope laser displays and secondary-cursor positioning, to name but three.
To enter Oscillator FX: press the LEFT mouse button over the Waveform icon.
To leave Oscillator FX: click on QUIT (lower left of screen, in a bit)
When you’re in Oscillator FX you can attach oscillators to secondary cursor position, primary cursor position, overall screen size and positioning, an animation sequence or a pattern-shape sequence. Mind you, just about all these functions can be better controlled directly from KML, but if you just want to do something simple without delving into KML then Oscillator FX can be used.

PATTERN MODE/EDIT



To enter Pattern Mode/Edit, press the RIGHT mouse button over the Pattern Mode icon.
To leave Pattern Mode/Edit, click the FINITO (Lower Right of screen).
This option allows you to choose between three pattern-generation modes: Decay, Line and Expandor. You can edit up to sixty-four patterns in each mode through this screen, and perform simple rotations on the patterns.

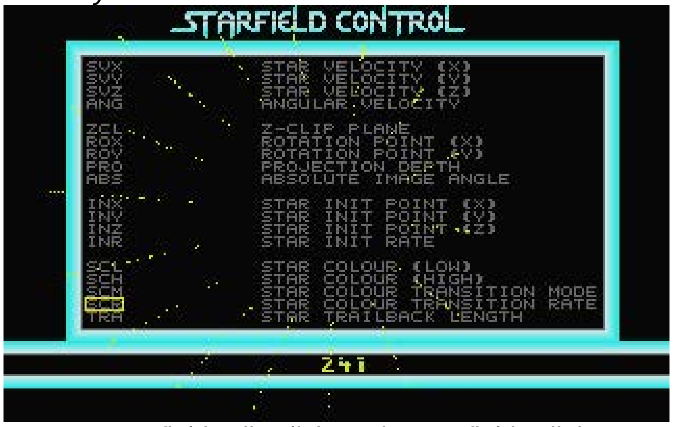
COLOUR COOKER



To enter the Colour Cooker, press either mouse button over the Colour Cooker icon.
To leave the Colour Cooker, click on Quit Edit (lower left of the screen).
The Colour Cooker is a useful device. With it you can define any of the 200 palettes available, set up to four rotation ranges, and if you desire attach up to three oscillators to the RGB within each range. You can construct smooth colourflows by just specifying key colours and using Calc Flow to generate the ‘inbetween’ colours.

STARFIELD EDIT

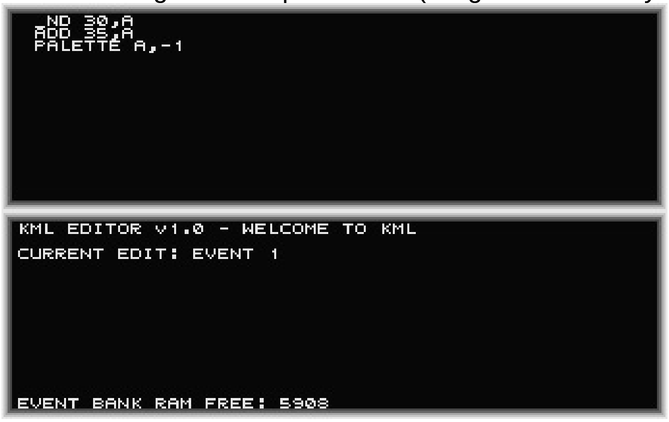
Note Starfield Edit available to users > 512K only – although you can still generate starfields from KML with only 512K



```
STARFIELD CONTROL

SVX     STAR VELOCITY (X)
SVY     STAR VELOCITY (Y)
SVZ     STAR VELOCITY (Z)
ANG     ANGULAR VELOCITY

ZCL     Z-CLIP PLANE
ROX     ROTATION POINT (X)
ROY     ROTATION POINT (Y)
PRO     PROJECTION DEPTH
ABS     ABSOLUTE IMAGE ANGLE

INX     STAR INIT POINT (X)
INY     STAR INIT POINT (Y)
INZ     STAR INIT POINT (Z)
INR     STAR INIT RATE

SCL     STAR COLOUR (LOW)
SCH     STAR COLOUR (HIGH)
SCM     STAR COLOUR TRANSITION MODE
SCR     STAR COLOUR TRANSITION RATE
TRA     STAR TRAILBACK LENGTH

241
```

To enter Starfield Edit: click on the Starfield Edit icon.
To leave Starfield Edit: press ESC (Starfield Edit is keyboard-controlled)
Starfield Edit is used for messing about with starfield generation. All the KML parameters govening generation may be viewed and altered, and the resultant starfield superimposed on the display, along with any symmetry you've currently got running.

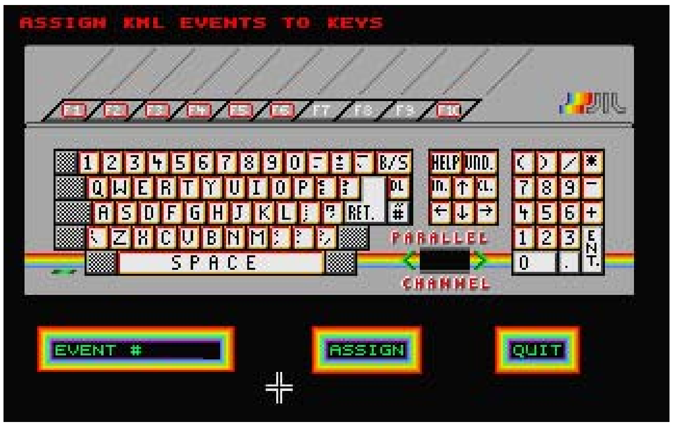
KML
This icon has two functions: Program Edit and Event Assign
To enter Program Edit: press the RIGHT mouse button over the KML icon.
To leave Program Edit: press ESC (Program Edit is keyboard controlled)



```
 AND 30,A
 ADD 35,A
 PALETTE A,-1

KML EDITOR v1.0 - WELCOME TO KML
CURRENT EDIT: EVENT 1

EVENT BANK RAM FREE: 5908
```

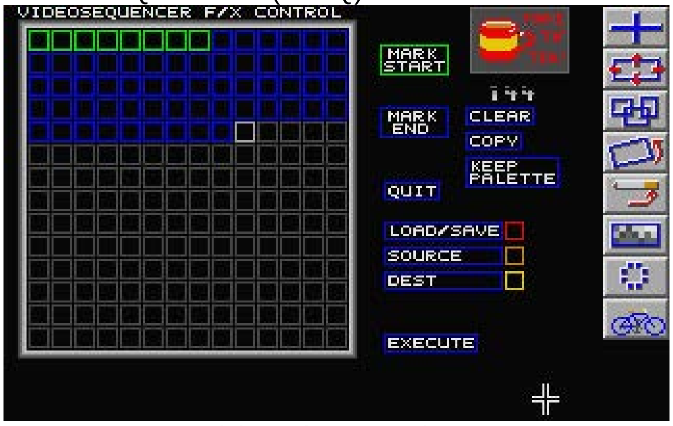
Program Edit is where you go to write your KML programs. The screen-based scrolling editor allows you access to the maximum possible 128 programs resident in memory. KML is the core of Trip-A-Tron. KML mastery will give you full control over the system.

To enter Event Assign: press the LEFT mouse button over the KML icon.
To leave Event Assign: click on Quit (lower right of screen)



Any of the KML programs (or ‘events’) can be attached to any of the keys on the keyboard. This makes it possible to completely customise the layout of your lightsynth keyboard controls. Specific parallel channels may also be assigned from this screen (KML is a parallel language, capable of running up to eight programs simultaneously)

VIDEO SEQUENCER (VSEQ)

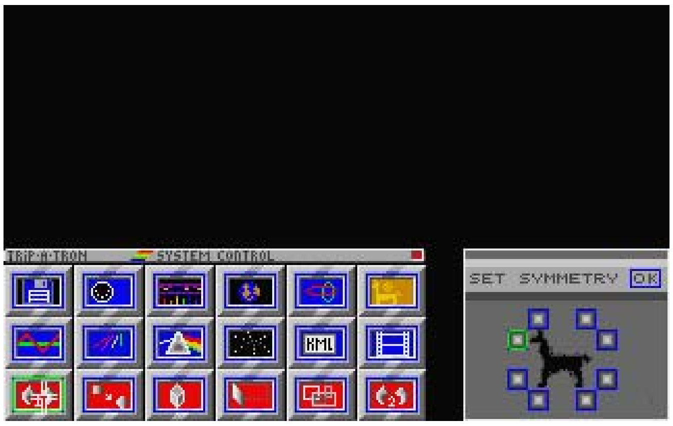


To enter VSEQ: click on the VSEQ icon.
To leave VSEQ: click on QUIT (slightly to right of centre screen)
The VSEQ is only of any use if you have 1 Megabyte or more, but the effects it creates are spectacular. VSEQ is basically a multi-frame framestore into which NEO, DEGAS or RLE files can be loaded. Once loaded, the frames can be previewed and subjected to a variety of manipulations: rotates in three dimensions, reflects, symmetry, pixellisation, filtering, texturing… Single frames can be treated, or whole animations. Completed sequences can be saved in the compressed .RLE (RunLengthEncoded) format.

The Bottom Row Icons: Plot Routine Control
These icons differ from the rest in that they're all switches as well as entrances to various edit screens. Clicking the LEFT mouse button on any of these icons toggles it on or off. An ON function is denoted by a green border.

Bottom-row, left-hand icon:

PRE-SYMMETRY SELECT:



To bring up Sym panel: press RIGHT mouse over the presym icon
To put away Sym panel: press OK on the upper-right of the Sym panel
This function allows te setting of Pre-Symmetry on the main plot routine

REMAP SELECT:
Note: this function only applies if you have a .MAP file in VSEQ memory
To enter Map Select: press the RIGHT mouse button over the Remap icon
To leave Map Select: click on OKAY (bottom left of MAP display)
Remap takes the X-Y co-ordinates of the points plotted and re-maps them into an arbitary surface defined by the currently-loaded .MAP file. Using this function you can map patterns or images onto spheres, disks, anything you care to generate a .MAP file for.

CUBIC REMAP
To bring up the Cube Face Select panel: press RIGHT mouse button over the Cubic Remap icon
To put away Cube Face Select panel: click on GROOVY on the Select Panel
This function replicates the central 200x200 pixel display area on one, two or three faces of a cube.

SHEAR
There is no EDIT screen associated with the Shear function. This function takes the display, reduces it in size, and shears it vertically so that it lies slanted across the left-hand side of the screen.

RESIZE
To enter Resize Edit: press the RIGHT mouse button over the Resize icon
To leave Resize Edit: click on he little tick (rightmost icon of the tiny icons at the bottom of the screen)
This function allows the replication and scaling of the display to fit up to 14 display windows definable from the Resize Edit screen.

POST-SYMMETRY
Controls exactly as for PRE-SYMMETRY except that the Sym panel affects Post-Symmetry instead of Pre-Symmetry.

COLOUR-CHANNEL CLUSTER
(icons at right of main icon bank)
These allow colour-multiplexing and are separate from the main Palette definitions. The buttons act as switches if the left mouse button is used; the right button takes you directly to Colour Edit. Each channel can hold a single, static palette; if more than one channel is switched on, the pallettes are multiplexed (allowing you to extend the ST palette range).
Have a wander around the various screens. Don't worry about the functions of each screen just yet – I'll cover everything in detail in the next section. When you're ready to proceed to the next section, every function will be explained in detail.

TRIP-A-TRON Detailed Function Descriptions

USING THIS SECTION

If you're new to Trip-A-Tron, I'd recommend starting your learning with the bottom row of icons, those governing the Main Plot Routine. Become familiar with these, then progress onto Colour Edit and Pattern Edit. By the time you finish with those two, you'll be ready to take a first look at KML. Mastery of KML is the key to complete control of the system. Beyond that you can go on to VSEQ handling, Sillyscope and Waveform control, and the rest of the functions. Each function or set of related functions has its own section in this Manual: feel free to delve at random if you're not into the sequential approach, although I'd recommend starting with the simoler functions if you're new to using a lightsynth.

TRYING OUT THE RESULTS

The blank area above the Control Panel is the **scratchpad area**. You can try out bits of pattern here just as a rough test. Starfields and Sillyscope displays don't appear here though, and no KML will run unless the panel is down. To leave the scratchpad and access the main pattern screen, just press ESC to lower the Control Panel. (ESC brings it back)

Bottom Row: THE MAIN PLOT ROUTINE CONTROL

Starfields, patterns and line effects are all composed of discrete pixels. The Main Plot Routine governs what happens to each pixel as it is plotted. The co-ordinates of the pixel are passed into the Plot Routine wherein the pixel may be transformed, replicated symmetrically, and finally spat out the other end and plotted.

The six icons in the bottom row control the function of the main plot routine, and thereby dictate the degree of complexity of the lightsynth pattern generation. The functions are applied 'left to right' looking at the icons. Thus, a pixel entering the routine encounters:

PRE-SYMMETRY: The pixel is replicated symmetrically according to the settings on the PreSym panel. EACH resultant pixel is then passed on to:

RE-MAP: The co-ordinates of each pixel passed to this routine are transformed onto an arbitary surface defined in a .MAP file. The resulting transformed pixel is then passed on to:

CUBIC REMAP: The pixel undergoes a series of scaling and shearing operations to place it upon the face of a cube. If more than one cube face is selected, the pixel is replicated the requisite number of times. Each resultant pixel is passed on to:

SHEAR: The pixel undergoes a simple reduction and shear operation and is passed on to:

RESIZE: The pixel is scaled to fit into resize windows defined in the Resize Edit Screen. Up to fourteen windows may be defined. The pixel is replicated once in each window. Each new pixel is passed, finally, to:

POST SYMMETRY: Another Symmetry operation, works exactly as Pre-Sym.

So, as you can see, a single pixel entering the plot routine can undergo a large amount of transformations and replication. Indeed, with everything running at maximum, a single pixel entering the routine could result in 8x3x14x8 – 2688 pixels! Such a gigantic plot overhead would result in really slow pattern generation – and while you may want that effect sometimes, you certainly won't want it all the time, so each of the six parts of Main Plot can be switched in or out, and its parameters controlled, by using the six bottom row icons.

PRE-SYMMETRY

Turn Pre-Sym ON and OFF by clicking on it with the left mouse button (the icon will be highlighted when it's turned on).

Edit the symmetry using the **Sym** panel. You get the Sym panel by pressing the right mouse button on the Pre-Sym icon. The Sym pannel will replace the colour channel cluster at right of screen. Click on any of the points of symmetry to turn them on or off ('on' points are highlighted in green). You can try out the symmetry on the Scratchpad display. Click on the QUIT button (upper right of the Sym panel) when you're done. REMEMBER – you won't see the symmetry if the Presym function is not itself enabled!

REMAP

Turn it ON and OFF with the left Mouse Button as usual. This functionality only works with a .MAP file in memory, so if one is not present an alert box will tell you so. Click on the alert boxes' exit button to clear the error.

If one or more .MAP files are in memory, pressing the right-hand mouse button on the Remap icon will bring up the Map Select display. The current map will be displayed (a grid will be plotted showing the shape of the map). If more than one map is loaded, you can switch between the maps using the left- and right-arrow icons at the bottom-left of the display. When the map you require is displayed, click on OKAY to leave Map Select.

CUBIC REMAP

Turn it ON and OFF with the left Mouse Button as usual.

Pressing the RIGHT Mouse button over the Cubic Map icon brings up the Cube Face Select panel, replacing the Colour Channel cluster. To select or de-select a cube face, click on the button on each face. A selected face is highlighted by a green button. When you've chosen the faces you require, click on GROOVY on the Face Select panel.

SHEAR

Turn the SHEAR function on and off with the left mouse Button. The shear command doesn't need any associated Edit screen.

RESIZE

Turn RESIZE on and off using the left mouse button as usual.

Pressing the right mouse button over the icon takes you to the Resize Edit screen. This screen consists of the following displays:

1. The Resize boxes themselves. Provided they're positioned on the visible screen, you'll see the boxes defining the areas into which the lightsynth display will be scaled. The **current box**, the one being edited, is highlighted in white.
2. The Priority Display. In the upper left-hand corner of the screen you'll notice a stack of 'corners', when the VSEQ 'resize and replicate' option is used to create multiple, stacked images, it draws first the image in the window at the 'bottom' of the resize stack (the 'bottom' window is the one whose 'corner' in the stack is closest to the upper-left of the screen). The current box's 'corner' is highlighted in white. You don't need to worry about Priority Display until you start to use the VSEQ.
3. The current box co-ordinates. These are displayed as four numbers in the bottom right of the screen. The first two numbers represent the x-y position of the ipper-left of the current box; the last two represent the size of the box in pixels, x and y.
4. The edit icons. The row of tiny icons at lower-left of the screen allows you to insert, delete, resize and centre the boxes, and to quit the edit screen.

Editing the Current Box

There are two ways to alter the size and position of the current box. The simplest way to understand is just to click on the box co-ordinate numbers at bottom right of the screen. If you click on, say, the tens value of the box x-co-ordinate, with theleft mouse button then 10 will be added to the value. Clicking with the right-hand button will decrease the value by 10. The same applies to the box size values.

Alternatively, you can use the Edit icons. These icons, from left to right, have the following functions:

1. **Select next box**. Highlights the next box in the list, making it the Current Box. Left mouse button steps forwards through the list, right mouse button steps backwards.
2. **Add a box**. Clicking on this inserts a new box 'on top of' the current box. The cursor will change into an upper-left box corner. Position the corner roughly where you want the box to go, then press and hold down the left mouse button and 'drag out' a box. When you release the mouse button, you'll be able to move the completed box around the screen. Position it where you want it, then press the left mouse button again. The box will be fixed in place, and will become the Current Box. You can fine-tune the box position and size by editing the box co-ordinates.
3. **Delete a box**. Clicking on this deletes the highlighted (current) box. You cannot delete the very last box left.
4. **Resize the current box**. Clicking the left button on this icon allows you to redo the 'drag out/move into position' stuff on the current box without going to all the hassle of deleting/re-inserting the box. Clicking the RIGHT button on this icon allows you to just move the box around, leaving the size unchanged; just click the button again to lock the box in its new position.
5. **Centre up current box**. Clicking the LEFT button on this icon centres the current box in X. Clicking the RIGHT button centres the current box in Y.
6. **Fine Adjust/Box X Position**: Clicking the LEFT button here subtracts one from the current box's X position. Clicking the RIGHT button adds one to the current box's X position.
7. **Fine Adjust/Box Y Position**: As for X position adjust, only alters Y position.
8. **Fine Adjust/Box X Size**
9. **Fine Adjust/Box Y Size**
10. **Quit**: When you're done editing the resize boxes, click on the tick to leave the edit screen.

POST-SYMMETRY

The final icon of the Plot Routine controls. Turn it off and on with the left mouse button. The right button bings up a Sym panel which is used in exactly the same way as that for Pre-Sym.

HINTS ON PLAYING WITH THE PLOT-ROUTINE FUNCTIONS:

Don't clog up the system with too many heavy functions – the system'll slow down so much you won't get a feel for what's going on. Don't use PreSym at all if you're just going for straightforward pattern effects. Try using PreSym, then a function, then PostSym and see how complex effects can be created. Just mess around and have a zarjaz time.

THE COLOUR-EDIT FUNCTION: Middle Row, Third Icon from Left
Pressing the mouse button over the Colour-Edit function takes you into the Colour Cooker, a powerful utility for generating both static and dynamic palette effects.
COLOUR COOKER: Main Display Features
The Colour Cooker consists of the following display areas:
1: The Colour Preview Area. This is located in the upper half of the screen. The 16 available colours are laid out in various configurations; the large boxes in the middle of the Preview Area can be used to select a colour to edit in the same way as their counterparts in the Control Area directly below them.
2: The Control Area. This area is in the lower half of the screen, here are located the individual Colour Select boxes (labelled 0-F directly above them are the Rotate Range Indicators (horizontal lines of yellow, red, green and blue, bracketing the selected ranges); to the right of the Range Indicators are the Range Select Buttons (a cluster of four buttons, Yellow, Red, Green and Blue). Lower down the Control Area you'll find the Calc Flow, Set Range and Quit Edit buttons; then the RGB Sliders, the Palette Number Select controls and the Rotate Speed/RGB Oscillators control box. Underneath everything you'll find the 'To Colour Channel' boxes.
Whew!
Don't panic. It's quite intuitive, really.
First, we'll tackle editing a static palette. Get into the Cooker. If the palette is dynamic (the colours are flowing) just use the Palette Number Select buttons to find a static palette (the buttons dial up any of the 200 available palettes. Press the << button to go down 10 palettes, < to go down 1 palette > to go up one palette, >> to go up 10 palettes). Palette 130 is empty, go there to try out the functions.
Okay, you have a static palette. Clear the the palette if it isn't already empty, by pressing the RIGHT HAND mouse button on the Calc Flow box. This will clear the whole range.
Select a colour (try colour 1) by pressing the left mouse button on the colour number, 0-F, you wish to alter. The selected button will be highlighted in red.
Now, with the button highlighted, use the left mouse button to adjust the red, green and blue colour sliders. As you do so, you'll notice the colour appearing in the Colour Preview area. That's groovy.
Okay, you've done colour 1. Using what you just learned, edit a different colour on colour #8. Then, leaving colour 8 highlighted press the LEFT mouse button over the CALC FLOW box The box will be highlighted in blue. Go to colour #1, the one you first defined, and click the left button on it. The program will calculate all the intermediate colours between the two hues you selected, and fill them in. You can use CALC FLOW between any two colours in the palette – useful for making smooth colour flows.
So now you can edit entire static palettes. Play with CALC FLOW for a while, then proceed to the next bit – setting up and animating colour rotation ranges.
To define a range, you must first tell the system which of the four roration ranges you want to use. Click on any of the four coloured Range Select buttons – Yellow, Red, Green or Blue. You'll notice the rotate range box highlight in the selected colour. Any adjustments you make to the rotate speed sliders and RGB colourflow controls will apply to the colours bracketed by the range indicator line of the equivalent colour.
Okay, you've chosen a range colour, Select colour 1, press the left mouse button on the SET RANGE button (it'll highlight in blue); then press the left button on colour #8. You'll notice that the Range Indicator line now spans colours 1-8, You can set ranges across any or all of the colours in the palette.
With a range selected, go to the ROT slider in the rotate range box. Using the LEFT mouse button, drag the slider to the left and right to set the speed of colour rotation. You can stop the range's rotation by pressing the RIGHT mouse button over the slider, which zeros it.
By selecting the differently-coloured rotate range buttons – Yellow, Red, Green and blue – and selecting a range, you can make up to four independent rotation ranges, each a different speed. You can even overlap ranges for special effects.
Play with this awhile until you're completely happy with it. Then move on to:
RGB colour oscillators. The colours on the Atari ST are made up of components of the primaries red, green and blue, eight levels of each. It's possible to generate complex, continuously-changing colour flows by attaching the r, g and b components of a colour to oscillators.
To use RGB oscillators, you must have some waveforms in memory (you can either use Waveform Edit to make up wave forms, or if you've been running the Demos, some waveforms will already be there).
Okay, here comes the example. Using what you've learned, set up a rotation range from colour 1 to colour F, and assign it maximum rotation speed, to left or right, it doesn't matter. Then, click the right mouse button in the 'Call Flow' box to clear out any colour you may have already defined.
Now, look at the other stuff in the rotate control box. Notice three buttons, red, green and blue, next to another slider. There will be a highlight around one of them, If it isn't the red one, click on the red button to highlight it.
Underneath the slider you'll see nine boxes, three of which are highlighted in red, green and blue. The first eight (square) boxes represent Waveforms 1-8; the ninth (oblong) button means 'don't use any

waveform at all (return a zero constant)'. Now, press the RIGHT mouse button anywhere over these buttons.

You'll see the sliders replaced with a picture of the three waveforms currently selected; red, green and blue, with red uppermost. Try using the LEFT mouse button on the waveform select buttons 1-8. Notice the red waveform changing shape as different waveforms are selected.

Return the red waveform to number 1 (the sine wave) by pressing the left mouse button on the leftmost waveform select button. Then, press the RIGHT mouse button anywhere over the waveform select buttons. The sliders will re-appear.

Use the LEFT mouse button on the slider next to the RGB select buttons to feed in the Red waveform. The slider controls the speed and direction of propagation of the red wave. Experiment. Notice how the pulsing red flowing through the palette changes frequency as you adjust the wave. Try using the Wave Select buttons to see the effect of different waves. Slow sines are best when you're learning about RGB colourflows, 'coz they're very obvious Clicking the RIGHT mouse button over the wave slider will halt the wave generation.

Try selecting the Green and Blue components and feeding in their waves on top of the red one. Experiment some.

Note: for more advanced users: when the waveforms are displayed in place of the sliders, you can phase-shift the current one by moving the cursor on top of the waveform display and holding down either mouse button to alter the phase.

Don't worry if the RGB oscillators seem a bit weird at first. As you become used to working with waveforms in general, you'll soon become used to them; and RGB colourflows can be exceedingly groovy!

COPYING A PALETTE TO A COLOUR CHANNEL: You can copy the current palette to any of the nine static Colour Channels by clicking on the appropriate 'To Colour Channel' button with the left-hand mouse button. The screen will flash blue to signal the transfer.

COPYING A PALETTE FROM A COLOUR CHANNEL: You can take the contents of a colour channel and copy them into the current palette. Click the RIGHT mouse button on the 'To Colour Channel' button of the colour channel whose colours you wish to take.

LEAVING THE COLOUR COOKER: You leave the Colour Cooker by clicking on the Quit Edit button. When you leave, the last palette edited becomes the current palette.

--- Colour Cooker Shortcuts ---

You can repeat a colour across a whole range. Go as if you were going to do a CALC FLOW, but press the RIGHT-hand button over the destination colour. All the colours in between will be filled with the source colour.

The RIGHT-hand mouse button on the CALC FLOW box will clear the entire palette.

THE COLOUR CHANNELS

A wee note-ette here to explain Colour Channels, as they're related to Palette functions. There are nine Colour Channels, and each Channel can hold just a static palette: sixteen colours. If no Colour Channels are selected (highlighted in green on the Control Panel), then pattern generation takes place using whichever of the 200 main palettes is selected. However, selecting **more than one** colour channel results in the open channels being *multiplexed* each 1/50 sec. By carefully selecting which Channels to multiplex, you can extend the colour range of your ST by creating 'false' colours made up of two or more 'true' colours being rapidly switched between. Try it --- maybe you'll like it – but remember to turn off any open Chanells if you want to see your main palettes.

PATTERN EDIT FUNCTIONS: Middle Row, Second Icon from Left

Click on the icon to get into the Pattern Edit screen. Look at the top of the screen to see which Pattern Mode is selected. If it isn't the Decay type, click on the 'Decay' box to select that type of pattern. I'll begin my explanation with those.

Examine the screen. You'll see at the top the Mode Indicators: Decay, Line and Expandor, with Decay highlighted. You'll also see the Current # display, with underneath the current pattern and a pair of arrow buttons. You use these buttons to change the current pattern, whether you're in Decay, Line or Expandor Modes. Use the arrows now, to select pattern number 1.

Decay-Mode Patterns

Decay-type patterns are the standard Colourspace pattern-type. Look at the screen and notice the following features:

1: The enlarged-view box, which, if you've been running the Demos, will contain a few pattern circles and a white dot. The white dot represents the centre of the pattern.

2: The whole-view box, in green on the right-hand side of the display. The pattern will be in there, along with a purple box representing the view selected in the Enlarged-View box.

3: The Current Level, Insert Level and Delete Level buttons.

4: The Mem Free figure. If you've been running the Demo, this won't be very much!

5: The Set Circle Point and Pixel Size controls.
6: Underneath the Enlarged View Box, the Next, Prev, Clear and Rotate controls.
Right. Before we do anything else, if you've been running the demos you'll notice that the Mem Free figure is pretty small, so let's do something about that. Press the CLEAR button. Well, that's a bit better, but let's make plenty of room: Use the Current # arrows and Clear button to clear patterns 1-4. Return to Pattern 1.
Now, to understand Pattern Edit, you need to know exactly what each pattern type is. Decay Type patterns consist of a number of 'levels'. Level 1 is the first and smallest level. When a Decay Mode pattern is plotted on the screen, here's what happens:
1: All Level One pixels are plotted, in their original colours, the first time around.
2: After a certain time (the Smoothing Delay constant determines this time) all Level One pixels are re-plotted **colour-shifted by –1**; then all Level Two pixels are plotted in their original colours.
3: After the smooting-delay-time, levels 1 and 2 are re-plotted, **colour-shifted by –1**, then all Level Three pixels are plotted in their original colours.
4: This process carries on in this fashion until all pixels of the final level have been drawn, Then, there is a wait of another smoothing-delay-time, and then all the pixels are unplotted.
So, a Decay pattern consists of a number of levels. The greater the number of levels, the longer the persistence of the pattern. Okay, let's do a Level One of a simple pattern. See the vertical row of boxes next to the enlarged display grid? The highlighted box represents the initial colour of any pixel you draw into Level One. Click on a colour that's more or less central, and, using the cursor and the righr-hand mouse button, dot two or three pixels into the enlarged-grid box.
Leave the pattern editor via the FINITO button, and try out the pattern in the Scratchpad Area. You'll notice that it's very fast – with only one level, there's hardly any persistence of the pattern. Re-enter Pattern Edit and click on INSERT LEVEL. You'll notice the pixels changing colour as they're colourshifted by –1. Dot in two or three extra pixels then quit the Editor and try again.
You'll notice that the pattern now trails back quite nicely as the delay introduced by adding a level takes effect. Go back to the Editor, do an Insert Level. Now, try clicking the left-hand mouse button anywhere in the WHOLE VIEW box. The purple square representing the Enlarged View will shift to centre on the area where you clicked. Dot in a few extra pixels and try the pattern once more. You'll notice that the pattern's starting to look a lot more like a normal Colourspace pattern.
You've now got the basic idea of how to build up a Colourspace trace. Just add levels and pixels, and test the pattern. Obviously, the more gubbins you add, the more inertia the pattern will have, so it's up to you to choose your patterns carefully according to whether you need a slow, flowing performance or a fast zappy one. By clicking on DELETE LEVEL you can remove a whole level of the pattern; you can edit levels already done by right clicking on the CURRENT LEVEL box (left mouse – down a level; right mouse – up a level). Pixels added or deleted will be on whatever level is showing in CURRENT LEVEL. You can delete an existing pixel by clicking the right-hand Mouse Button on it. If you place a pixel on top of a previously-placed one, the previous one will be replaced.
You can also alter the pixel size of individual pixels within a pattern. Before you insert the new pixel, use the arrow buttons around the PIX SIZE X and PIX SIZE Y buttons to choose the new pixel size, up to 8x100! Then, just insert your pixels as normal (bear in mind that large pixels have a larger speed-overhead)! Yeah. Build yourself a five- or six-level Decay pattern, not too complicated, then when you're satisfied, re-enter the Editor. Click on the NEXT box, which will take you to pattern #2. This should be empty, you cleared it earlier, remember? Okay. If you pressed PREV at this time it would take you back to pattern #1. However, press PREV with the RIGHT mouse button and, provided the current pattern is empty, the previous pattern shape will be *copied* into the current one. Do that, and see it happen. OK? Now click on DO ROT8 and you'll see that the pattern shape has been rotated to the right 16 Yakly degrees. (Yakly degrees are different from normal ones. There are 360 normal degrees to a circle, but Yak thinks 360 is a nasty number to deal with, so in all Trip-A-Tron rotation ops there are 256 Yakly degrees to a circle, 256 being a Holy Number and a lot more fun to work with).
You can change the number of the Yakly degrees per rotation by using the arrow keys next to the DO ROT8 value. By building up a succession of partially-rotated patterns and animating through them with the KML SET n, PATTERN command, you can make all sorts of groovy spinny bits and pieces like the ones in the Demo.

Fine. Now you can make Decay patterns, and you know how to do Rotates, and all about Yakly degrees. Now, Yak will teach you about EXPANDORS. I'm doing Expandors before Lines because they have a lot in common with Decay patterns when it comes down to editing.
Select EXPANDOR at the Pattern Mode display at the top of the screen, then select pattern #1. Before you CLEAR the pattern to begin editing, take a look at the new features on the display. All the LEVEL

controls have disappeared – Expandors have no levels – and you've got Rotate Mode, Lifetime and Velocity Vector controls added. To understand what's going on, let's look at what an Expandor really is.

When an Expandor is plotted, what happens is this:
1: The pixels are drawn, at their starting positions and starting colours (i.e as you placed them in Expandor Edit)
2: There is a delay of one Smoothing Delaysworth. Then, the old positions of all pixels are unplotted, and the Velocity vector associated with each pixel is added to that pixels' position; the pixel is re-plotted, colourshifted by –1.
3: This is repeated for a count equal to the Lifetime value of the Expandor.
Expandor effects tend to be very liquid/fluid/flowing, and quite different from Decay effects.
Clear the current Expandor, select the base colour, Go to the Velocity Vector box, and using the left-hand mouse button within the box, choose a good long vector. Then go and dot a pixel into the Expanded View box. You'll notice a little 'tag' coming off the pixel lying there – that's the velocity vector. Quit the editor and go try it out.
When you're happy playing with the Expandor, go back to the Editor and add a couple more pixels. (Expandors are economical on memory; you get a nice big pattern from two or three pixels' worth of pattern definition, and indeed they become very slow if you use many more). Give each pixel a different Velocity Vector and study the effect.
The Next, Prev and Do Rot8 functions behave as for decay pattern-types, except that the Rot8 command can be set to affect only the Velocity Vectors of a pattern, or only the position of the pixels, or both Vectors and Position (use the left mouse button on the Rotate Mode boxes to highlight in green the rotate mode you desire).
Try adjusting the LIFETIME variable – this sets the duration of the Expandor's life. Very long Expandor lifetimes will result in slow plots. Up to 10 is usually the most feasible setting before it all gets too sluggish.
Oh, by the way, the Set CentrePoint command: this applies equally to Expandors and Decay types. Rotations are usually centred around the middle of the pattern. You can change this by selecting Set Centre Point, then clicking on the new rotation-point position in the Whole View box. Rotate commands will subsequently be centred around the new point. You can bypass pressing Set Centre Point by just clicking the RIGHT mous button in the whle View box.
Decay and Expandort types sussed – now on to the simplest pattern type, Line Mode. Use Mode Select and choose LINE.
Line Mode is entirely different from the other two pattern modes. When you draw in Line Mode, the program attempts to draw a line between the Primary cursor point and the Secondary cursor point (if the Secondary Cursor is turned off, this will be the point (160, 100) – the middle of the screen). The line is drawn with a 16-colour line mask determining the colours of the pixels in the line. Select line #1 and press CLEAR.
To set a pixel in the line, click the left mouse button in one of the line mask boxes. The box will fill with colour. If you continue to press the left button, the pixel colour will advance by +1. (If you press the right button, the pixel colour will decrease by –1). Fill in as many of the lines' pixels as you wish (the more pixels in the line mask, the slower and more colourful the result). Drop back to the Scratchpad to try out your Line Mode pattern.
Returning to the Editor, you'll notice a Maximum Line Length variable, with arrow keys for you to adjust it. This determines how much of the line is actually drawn. A high Maximum Line Length means that the line between primary and secondary cursors will nearly always be complete. With a low Maximum Line Length, only a little nubbin of line will be drawn out from the primary cursor towards the secondary cursor). Experiment with the value – try the effects. Note – to speed up your editing, when when you use the Line Length arrow boxes: LEFT button on an arrow adds/subtracts one from the value; RIGHT button held down continuously increases/decreases the value.
You'll notice little arrow keys underneath the Line Mask display – these can be used to rotate the whole set of Mask colours left or right.
Prev, Next and Clear all work in exactly the same way as for Expandors and Decay patterns. There is no MEM FREE value because all the memory needed for 64 Line Mode patterns has been pre-assigned; the memory used by editing never gets bigger or smaller, so you've no worry about running out of line RAM.
The only other feature left to mention in this Pattern Edit section applies equally to all three types of edit screen: the Preview Colour Channel buttons at the bottom of the screen. These buttons can be used to change the palette from those of the Edit Screen to those allocated to any of the Colour Channels. Just hold down either Mouse button over these boxes, and the relevant Colour Channel palette will be displayed so you can see roughly how your patterns and lines will look. Release the mouse button to restore the proper palette.

Now we have looked at palette editing and pattern editing, you'll probably be experimenting and producing patterns and palettes you want to keep. So it's probably a good time to take a look at one of the more important Trip-A-Tron screens: the Disk Access screen.

DISK ACCESS: Top Row, Leftmost Icon

To enter the Disk Access screen, click on the Disk icon. The Disk Access screen will rise up, and you'll see a rather groovy File Selector screen (I remember all that faffing about in Colourspace One with commands like 'undo-M-A' and not a directory in sight, and the groovy File Selector screen is by way of saying 'sorry' to all those people who had to suffer!)

Here's how it goes:

At the top of the screen are the letters A-J. To select a drive for LOAD and SAVE, just click on its drive letter. The drive letters of nonexistant drives will, of course, bring up no directory, nor should you attempt to save files on them!

When you click on a drive letter, the drive's root directory will be listed in the Directory Window (the lowest half of the screen). The root directory is the first layer of the disk, and any folders will be listed, along with the files. To 'go into' a folder, you just click on the folder's name in the Directory Window. You'll see the pathname being built up in the Pathname Window and the new directory will be built and displayed.

The filetypes listed depend upon the current setting of the *extension type*. Trip-A-Tron recognises files of twelve different formats, and each format is denoted by a three letter extension to the filename, as for example GOAT.NEO or SHEEP.BAA.

You set the current extension type, and thus what type of Load or Save to do, by clicking on the relevant Extension Type Box. These boxes lie directly under the drive select letters.

Changing the extension type causes a new directory to be generated, with folders and files of the appropriate type being listed.

The twelve extension types, and their meanings are here listed:

.NEO	This is probably familiar, as it denotes a NEOchrome picture file.
.PI1	Equally familiar, this denotes a DEGAS Low Resolution picture file.
.RLE	This is reserved for Run-Length-Encoded picture files generated by Trip-A-Tron. Run-Length-Encoding is a simple data-compression technique which means that most pictures saved in this type take less disk space than their Neochrome or Degas equivalents.
.KML	This is for KML program sets.
.MAP	This is for screen re-mapping files. Loading a .MAP file makes the REMAP option available on the Control Panel.
.PAT	This is the extension under which all pattern definitions are saved.
.PAL	This is for palette block saves and loads.
.YAK	This is the extender for RLE-STASH compacted animation sequences.
.PRE	This is the extender for Global Variables and settings.
.WAV	This is the extender for Waveform definition files.
.EWE	This is the extender for Macro Load files.

Loading and Saving Stuff – General Stuff

This is GENERAL stuff. It applies to all types of load and save operations, regardless of filetype.

Firstly and most obviously, be sure that you've prepared properly for your SAVE. This is important especially in VSEQ operations where you have to mark off a range of frames to be SAVEd.

The sequence to LOAD and SAVE is almost identical whatever you're doing. If you're just about to SAVE, have a formatted data disk in the drive. (If you change the disk in the drive, clock on the drive letter to tell the system you changed the disk. The directory will be updated). You can save all kinds of data files on a standard Atari formatted disk, but I recommend keeping separate disks for the different data types while you're working on a project, then putting all the files together on a single disk when you're ready for a display.

To LOAD a file: select the proper drive letter and extender type, and go through the directory, down folders if necessary, until the filename of the file you want is in the directory window. Click on the filename in the directory window to transfer it into the Filename Window. Then, just click on LOAD to load a file in.

To SAVE a file: If you're saving over a previous copy of a file, follow exactly the same procedure as for LOAD, but click on SAVE at the end, instead of LOAD. If you're saving a new file, go through the directory and folders etc. until you come to the place where you want to save the file. Then click INSIDE the File Name window. The cursor will disappear, and you can type the proper filename into the filename window. DELETE works, so you can take out unwanted *'s and .'s and typos. You don't have to type the extender; it's tacked on automatically by the SAVE routine. Filenames can be up to eight letters long. When you've typed the filename, pressing RETURN re-activates the mouse cursor; you can then go click on SAVE, which will do just that.

If there isn't enough room on your disk, an alert box will tell you so. Try again with a fresh disk or on a new partition if you're a hard drive user.

Special Cases of SAVE and LOAD: For some filetypes, there are some considerations to be taken into account when SAVEing and LOADing.
1: Multiple picture SAVEs and LOADs
When you're loading or saving a whole bunch of frames for VSEQ, there's a slight difference in the way filenames are used. You can use up to 6 letters in a MultiSave/Load filename. Set up the filename as for a normal SAVE or LOAD, but edit the filename field and tack a '*****' on the end. For example, if you had sixteen frames marked for a multisave in VSEQ, you would put GOAT* in the filename window with (say) a .RLE filetype. The SAVE would produce files called:
GOAT00.RLE
GOAT01.RLE
GOAT02.RLE
..
GOAT15.RLE
To re-LOAD these files, you'd follow the normal LOAD procedure until you can to the point where you had
GOAT00
In the Filename box. You'd alter this to
GOAT*
And press LOAD to load the frames.
SAVEing frames from VSEQ requires absolutely that a SAVE range be defined in VSEQ (see VSEQ section for details). LOAD would like a range, too, but if none is provided it will load from the last frame loaded upward.

LOADing and SAVEing Sequencer Files
LOADing a Sequencer file takes up to one frame of VSEQ memory. When you go to LOAD such a file, an alert box will appear to confirm your decision. You must have used the Sequencer before a Sequencer save operation can be used, too!

LOADing .MAP Files
.MAP files take up to eight frames of VSEQ memory, so upon loading these files in an alert box offers you the chance to abort the load. If a map file is already in memory, the same alert box appears: clicking on BOTTLE OUT then loads the selected file over the existing Map, replacing it; clicking on DIM SWEAT takes another eight frames of VSEQ Ram and puts the second Map in that. You can then switch between the Maps from the Control Panel or KML.

Other Things You Should Know about the Disk Screen
You can edit the pathname box in the same way as the filename one, but you should seldom need to, as pathnames are constructed automatically as you go down through the folders.
To **get out** of a folder you're in, press the **backward-facing arrow** icon (between Save and Make Macro) to retreat one step back up the directory. You can get back to the root directory anytime by clicking on the drive letter.
To quit from the disk screen, click on the little sheep with 'BAA' written next to it, the one standing on a little patch of grass next to the Make Macro button.

Other Functions of the Disk Screen: RLE Stashes and Macro_loads
RLE Stashes are special types of compressed animation files. The way they work is, basically, you define a LOAD range in the VSEQ, and then come to the disk screen and select the .RLE fileextender, just as for a normal .RLE load. Then, click on the TO RLE STASH icon. While this icon is illuminated, all .RLE files are loaded, then placed in compacted format in the space allocated. From there they can be decoded and animated using the KML UNPACK command. RLE stashes thus formed can be SAVEd in a lump as .BAA files. Loading a .BAA file creates an optimised RLE stash in VSEQ memory. Using this technique, it's possible to fit, say, fifty framesworth of animation in twenty framesworth of space, depending on the density of the picture data.
When you activate the TO RLE STASH icon, you'll get a requester asking you if you're sure you want to lose the VSEQ memory.

Macro Load
The MACRO Load facility is very useful. By using Macro Load, you can make the system boot up with your own default files already in memory, or load all the files for a complex display with only two LOAD commands – one for the .EWE file and a Macro Load command.
.EWE files – Macro Load files – are really just lists. The list contains the locations and names of the files to load. For example, if you have the Demo files loaded, going to the disk screen and selecting .KML as the extender, click the RIGHT hand mouse button on the Make Macro button. You'll see in the directory window something like

```
        MACROLOAD FILES FOR FILETYPE .KML
A/DEMO
```

Click on OKAY to restore the directory.
All this is is an instruction to the computer that during loading looks for a file DEMO.KML on drive A.

Building Your Own Macro Load Files

This process is really quite painless. First, you need an empty Macro Load buffer. The buffer is empty if you boot up on a drive with no DEFAULT.EWE file. Don't worry if you had a DEFAULT file in though – just use the disk screen to LOAD the file EMPTY.EWE provided on the Trip-A-Tron disks.

With an empty macro Load buffer, the next stage is to place the disk or disks containing the files you want to Macro Load in the drive(s). You must ensure that at the time the Macro Load takes place, these disks are in the same drives as when you made the Macro Load file.

Next, for each file category, go exactly as if you were going to manually load the files, but instead of pressing LOAD, press the MAKE MACRO button with the LEFT hand mouse button. You'll see the file added to the Macro Load list for that filetype. (You can have up to eight Macro Load definitions for filetypes .NEO, .PI1, .RLE and .MAP).

If you make an error, don't worry, you can examine a Macro list without adding to it by pressing the RIGHT hand mouse button on the MAKE MACRO button. You have the option to delete the last file added, so you can zap erroneous entries.

When you've made macros for all the file types you need, do a SAVE in filetype .EWE to create the Macro Load File. Then, whenever you need to reload all those data files, all you have to do is load the .EWE file and click on LOAD MACRO to load them all in. Try it. Bung the Trip-A-Tron disk in drive A, then use LOAD to load in the Default.Ewe file. Then click on LOAD MACRO and see the suckers load. It's a neat trick which saves a lot of time loading a complex display.

If you save your Macro Load File as DEFAULT and place it on a Trip-A-Tron boot disk, you can make the system load in your stuff automatically at boot-up. (Hard disk users – the DEFAULT file must be in the same directory as the TRIP.PRG file.)

Examples of using the Disk Screen

1: SAVEing a bunch of paterns and palettes on drive A

Bring up DISK screen; click on drive letter A

Use direcory to get to where you want to save files

Click on extender .PAT

Make a filename in the Filename Window

Press SAVE. Click on .PAL

Make a filename. Press SAVE. Easy, huh?

2: LOADing thirteen frames from drive B into VSEQ (assume the frames are called HORSE00.RLE, HORSE01.RLE, etc.)

Place disk with files in B, bring up the disk screen, click on B to get the directory

Go through directory until you see HORSE00.RLE

Click on the name in the directory

Edit the filename from HORSE00 to HORSE*

Press LOAD. No hassle.

Exercise for Hard Disk Users: Use what you have learned to transfer the Demo fukes onto hard disk and make them auto-boot. The demo uses files .KML, .PAT, .PAL and .WAV, and each chains the other.

General Notes about the Disk screen:

In macro load, the VSEQ is cleared if any files in the Macro Load request a place in it. Frames are loaded from frame zero upwards; sequencer, map and RLE-stash data grows downwards from the top of VSEQ.

Macro_load can be initiated from KML via the MACRO_LOAD command. The command causes the .EWE definition from the currently-loaded macro_load buffer to be loaded, then all the files listed therein loaded subsequently.

When a DEFAULT.EWE file is found at boot-up, all the files are loaded before the Conbtrol Panel comes up. Immediately upon lowering the panel, either by hand or via a MIDI WAKEUP request, KML program #127 is executed. In this way remote machines in a midi network can load all their data and set themselves up without needing a keyboard or mouse input.

Well, now you can LOAD and SAVE anything you want. Time to go on to the next section… the entirely useful…

WAVEFORM CREATION and OSCILLATOR EFFECTS screens (Middle row, leftmost icon)

Waveforms are central to many of Trip-A-Tron's functions, like RGB oscillators and SillyScope displays. Some example waveforms are provided in DEMO.WAV, but you'll eventually want to try some of your own. First, we'll deal with the Waveform Editor.

Entering the Waveform Editor

Click on the Waveform icon with the RIGHT mouse button to gain the Waveform Editor screen. Take stock of the screen when it comes up; the central feature being the two large waveform displays filling the screen.

At the top of the screen, we have, from left to right:

The **waveform speed** buttons, left and right arrow buttons. Where the waveforms are to be used on the Oscillator Effects options, you set their speed from these buttons. The current speed of the waveform is shown by the colour running through the wave trace in the upper half of the screen. Click on these buttons to increase or decrease the speed. (Many of the functions which use the waves have their own, independent wave speed controls, so you can often ignore the Waveform Speed when editing a waveform).

Slightly to the left of centre are the three **waveform run** types: the icons represent, respectively: **Continuous Run, Bounce** and **Gated** mode. Again, these settings are unique to Oscillator Effects, and can often be ignored for general waveform use. Here's what each setting means:

In **Continuous Run** mode, the wave starts again from its start point when its end point is reached.

In **bounce** mode, the wave starts from its start point, and upon reaching its end point it is propagated **backwards** until the start point is reached again, at which time the wave reverses again and the cycle continues.

In **Gated/One Shot** mode, the wave remains at its start point until it is triggered by a KML GATE command; once triggered, it runs through once and then shuts off until GATEd again.

Now we have the Oscillator-Effects specific stuff accounted for, we're ready to take a look at the screen in more general detail. The two waveform displays contain the **Current Wave** in the top half of the display, and the **Source Wave** in the bottom half. You get to build waves by adding, subtracting, or modulating the Source Wave and the Destination Wave.

Check the upper left of the screen: the Waveform Number icon, followed by a pair of arrows and a number in a box. That number is the Current Waveform Number. You can have eight Waveforms in total; clicking on the arrows gets the next/previous Waveform, depending on which arrow you press.

Try it – dial up all 8 waveforms and see what they look like!

When you're done looking at the waveforms, get Waveform #1 back. Look to the left of the Waveform Number controls: you'll see the FROM icon, followed by another pair of arrow keys and a number. Using these controls, you can put any of the the eight defined waveforms in the Source Wave buffer, instead of the regular sine, saw or square wave you usually get there. Thus, you can copy one wave to another, or set up more complex wave additions than can be done with a regular wave. To get one of the other waves into the Source Wave buffer, dial up the number of the wave you require in the From number box using the arrows, then press FROM to copy the wave into the Source Wave buffer. Try this a few times, just to get the idea.

Okay, let's make a waveform.

Select Wave Number 1, Look down at the bottom of the screen. At bottom left there's a bunch of icons, three with a picture of a wave on each, and another pair of those funny little arrows. The three waveform icons represent the **source waveform type**. Assuming you don't want to use one of the other waveforms as the Source Waveform by using the FROM button as described just now, you'll have to start out with a basic sine, sawtooth or square wave. Click on the leftmost icon, the one with a sine wave on it, and – surprise – you'll get a nice fat sine wave in the Source Wave window. Try the Sawtooth and Square wave icons and see their respective waves appear as the Source Wave too. Pretty obvious.

Adjust the frequency of the Source Wave by using the funny little arrows icon – the top arrow decreases the frequency, down to 1 cycle; the bottom arrow increases the frequency. Play with this for a bit.

When you're done playing, look at the next group of five icons on the bottom row. They're **Waveform Manipulation** icons, and with them you can further modify the Source Wave before you add it to the Current Wave. Here's what these icons mean, and do:

(left to right)

Decrease Wave Amplitude: Used with the LEFT mouse button, this function halves the amplitude of the Source Wave. If used with the RIGHT mouse button, one is subtracted from all the values in the Source Wave.

Increase Wave Amplitude: LEFT button – DOUBLE the amplitude of the source wave. RIGHT button: ADD one to the entire Source Wave.

Rectify Wave: ANY button – makes all the negative parts of a wave, positive.

Negative Wave: ANY button – inverts the source wave.

Phase Shift: LEFT button – Phase shift by –1 Yakly degree; RIGHT button – Phase shift by +1 Yakly degree.

So generally fiddle with the source wave a bit – if you get in trouble just click on one of the three Waveform icons to get a pristine Source Wave. When you've fiddled, turn the bottom-rightmost icon: the **Add** button. This does the business of adding the Source Wave to the Current Wave.

The Add button operates on that portion of the total Current Wave which lies between the two Range Markers (see the bit in a mo about the Current Wave for more info on these; they're usually set to encompass the whole Current Wave anyway).
The Add button has two functions; to make the Current Wave by adding in the Source Wave; and to zerp the Current Wave. LEFT mouse button adds the waves; RIGHT mouse button zaps the Current Wave.
Let's try it. Cook up a suitable Source Wave. Click the RIGHT mouse button on the Add button – the Current Wave goes flat. It's been zapped. Now, click the RIGHT mouse button on ADD. The source wave is added to the zeroed Current Wave, and a copy of it now becomes the Current Wave. You just made a waveform. Cook up another, different Source Wave and add that in, too, just to see what it's like. If you don't like the result, you can click the Undo-Llama (the upper-left-hand-llama with 'Undo' written beneath it) to undo the last operation.
Experiement. Experiment. Like I always say it's the best way to learn.
Now, on. You've seen how to make a new Current Wave by adding the Source Wave to the Current Wave. There is one other way of merging Source and Current waves: **modulation**. Try this to see the effect of a Modulated wave:
Clear the Current Wave. Dial up a sinusoidal Source Wave of, say, eight cycles, and add it to the Current Wave. Then, go right back down to a single cycle of sine wave in the Source Wave, and click on the **Modulate** Llama (that's the lower-left-hand-llama).
Observe the effect on the Current Wave.
The way modulation works is this: the **absolute** values of the Source Wave are used as an envelope to modify the amplitude of the Current Waveform. The result is a Current Waveform with its own frequency, but **amplitude-modulated** to the shape of the Source Wave. Don't worry too much about how or why this works; just enjoy it. Try Modulating with other waveforms. You can produce some truly zarjaz quite easily with this technique.

Operations on the Current Wave

Examine the icons underneath the Current Wave. The central cluster of the five icons will be familiar to you – they're the same functions for Waveform Manipulation as those you've already used on the Source Wave. They behave in exactly the same way as the Source Wave functions **except** that they operate only upon that bit of Current Wave between the Range Markers.
Ah, yes, the Range Markers. I've touched on these briefly before; now for the full explanation.
Look at the four icons to the right-hand end of the Current Wave manipulation buttons. Look also at the two waveform displays – notice that at each end of the Current Waveform display there's a white bar. The four icons can be used to move these bars in order to bracket a piece of the Current Waveform. This is easy to do; just remember that the < and << buttons always move the left-hand (Current Wave Start) delimiter, and the > and >> buttons move the right-hand (Current Wave End) delimiter. The double-arrow buttons move their respective cursors faster than the single-arrow ones. The icons themselves do not control which way the lines move; the mouse buttons do. Pressing with the left-hand mouse button will, naturally enough, move a delimiter to the left; pressing the right-hand button moves a delimiter to (you guessed it) the right. (well, what did you expect – that the right –hand mouse button rotates the delimiter through the fourth dimension into Hyperspace?)
Have a go at delimiting bits of Current Wave, then I'll tell you why you should want to do it anyway.
Some Good Reasons To Delimit Bits of Current Wave:
1: The I-don't-need-a-whole-waveform reason: In all wave functions except SillyScope, you can create waves which consist of less than 256 values. This is useful for creating the types of waveforms which drift gently in and out of phase. 'Short' waves are created just by moving the start and end delimiters to bracket the bit of wave you want.
2: The I-only-want-to-change-a-bit-of-this-waveform reason: Addition and modulation of the current wave, and all the current wave edit icons, only operate on the bit of current wave between the two delimiters. Thus you can isolate and work on just bits of a wave.
3: The There's-a-couple-more-functions-you-can-do-with-delimited-waves-type reason: Examine the two little icons on the very left, underneath the current waveform display. These two icons are quite useful; here's what they do…
The leftmost icon of the pair is the **Expand Waveform Segment** icon. Clicking this will expand the piece of waveform between the delimiters to fill the whole Current Wave area, and re-set the delimiters to the extreme ends of the wave. Using this functions you can cut out and 'enlarge' bits of the waveform.
The other icon of the two is the opposite function: The **Contract Waveform** icon. This reduces the total Current Wave and squashes it up to fit into the space between the two markers. To leave the Waveform Editor, click on the Quit-Llama (that's the upper-right-hand-llama).
You now have the information to create and use Waveforms as complex as you desire. You can make up to eight Waveforms and SAVE them as .WAV files from the Disk screen. It's time to look at some simple applications of waveforms.

OSCILLATOR EFFECTS/CURSOR CONTROL SCREEN:

To get into the Osc FX screen: press the LEFT-hand mouse button over the Waveform icon.

Bring up the Oscillator FX screen and take a look at it. This screen is used for two purposes: attachment of certain system functions to the Waveforms generated in the Waveform-Editor, and management of pattern-generation cursors. We'll look at the simplest of these functions, cursor management first.

In default Trip-A-Tron mode, you have a single cursor; you move it with the mouse, and when you press the mouse-button, pattern-generation ensues from the cursor point. Actually Trip-A-Tron has **two** cursors: the Primary Cursor is the on you're used to, and the Secondary Cursor, which can be added at will for effect.

The secondary cursor is denoted by the notation X' and Y'. Look at the Oscillator FX panel and clock the two buttons, X' and Y'. and the stack of three buttons next to them. Those three buttons determine whether the Secondary cursor is on, visible and in what mode it is. If all three buttons are Off, click on the ON button. Both ON and VIZ will highlight. Just for now, click on TRAIL, too. When all three buttons are highlighted, leave the FX panel and go to the Scratchpad.

Don't generate any pattern just yet: just move your cursor about in the Scratchpad and notice that it now has a partner which follows it at a respectable distance . (This respectable distance is governed by the Secondary Point Trailback Length variable, set by one of the sliders in the Global Variables screen).

Now try generating some pattern. You'll notice (if you're not in Line Mode) that the patterns appearing at the secondary cursor are a slightly different colour. This is because all Secondary Point patterns are colourshifted by +8 relative to their Primary Point counterparts.

If you choose Line Mode and the secondary cursor is enabled, the lines or segments of lines drawn will be between the two Cursors.

Go back into Osc FX and look at the cursor control buttons. You can turn ON or OFF both Main and Secondary cursors from this screen, or just make one or other of the cursors invisible (use te mouse on the VIZ buttons; when un-highlighted, the cursor will not be plotted. This is useful in performances where you don't want the cursor to appear – but you'll have to have a good 'sense of cursor' not to get lost (although there are options to help you – see the Global Parameters section).

Experiment with the Cursor Control buttons, but leave the Secondary Cursor on TRAIL for the nonce. When you're happy. Proceed to the next section:

Oscillator Effects

The Oscillator Effects on this control panel represent the very simplest level of Waveform usage. Results obtained from here aren't as complex as those which can be achieved by clever KML programming, but they serve well to illustrate Waveform effects. I think the best way to learn about these effects is by way of a worked example.

Use the Disk screen to load the file DEMO.WAV.

Now, go into the Waveform Editor. Select waveform #1, and click on the CONTINUOUS RUN icon. Use the Wave Speed icons to get the wave going at maximum speed. Waveform 1 is going to serve as the X-waveform in our example.

Now, select Waveform Four and set the wave for Continuous Run, top speed. This is to be the Y-wave. (Remember in the Waveform Edit section I mentioned that all oscillator FX waves need their speed and run-type set up first on the Waveform Editor? Well that's what you just did. Don't worry though, you can still use the waveform for other functions).

Quit the Waveform Editor and go into Oscillator FX. Click in the X' button to highlight it. Notice 'OFF' appears highlighted in the BASE section of the control panel.

Oscillator effects require various pieces of setup info: namely, a **base**, a **source**, and a **scale**. The **base** is simply the midpoint around which oscillation will take place; the **source** is the number of the Waveform to be used; and the **scale** is the range covered by the oscillator during its oscillations.

So, for our X' wave, the **base** is OFF, which means the effect is disabled. The **base** can come from the current Mouse X or Mouse Y position, but for the purposes of demonstration, we'll use a constant base. Click on **constant**.

Now, we still need to tell the system **what** constant to use. I want the effect to centre about the point X=160, the middle of the screen. Notice the empty box next to the highlighted constant. That's a number box; to place a number therein, just click on the hundreds, tens and units positions in the box to get the appropriate digits. LEFTmouse clicks in a digit area adds 1 to that digit: RIGHT mouse clicks subtract 1. Straightforward, Doesn't sound it but is. Use the mouse to dial up the constant '160' in th ebox to prove my point.

Look at the SOURCE box when you've done that. When you activated the Base, the Source box immediately highlighted the default waveform btton #1. Since Waveform #1 is the one we've already set up, you need to make no more further adjustments to the Source setting.

Ignore the LOGIC box for now, we don't need it at the moment. Go right to the end, to the SCALER box. I want this effect to cover the whole X dimension of the screen – that's 320 pixels – so fill in the number 320 in the Scaler box using mouse clicks as usual.

Finally, ensure that the Secondary Cursor is on and visible, but de-select TRAIL.
Quit the Osc FX screen, and press ESC to lower the control panel. Look at the secondary cursor – you'll see it oscillating gently from side to side on the screen. It's under the control of the sine wave on Waveform One, centred around our Base of 160, to an amplitude of 360. You've seen that it works.
Okay, now let's see if you learned properly. Select the Y box and assign the following parameters:
A CONSTANT BASE of 100
A SOURCE of Waveform 4
A SCALER value of 200
Leave the FX screen and observe the results.
Now you have a basic ide of how to set up an Oscillator FX mode. I'll explain what the other modes are, and then it's largely up to you to experiment.
Look at the PRIMARY INPUT cluster. These four OSCFX settings all apply to the Primary Cursor. You can, if you wish, have both Cursors entirely under oscillator control. There are two X settings and two Y settings – the actual point and an offset from that point. Combining these four FX can produce weird results.
The SECONDARY buttons X' and Y', govern the secondary cursor and have already been explained.
The SIZER allows you to link the oscillators to overall screen width and height (first column of two icons) and overall screen positioning (second column of two icons). If these FX are active, all pattern plot is scaled and translated according to the state of the oscillators at plot-time.
The VSEQ lets you hook up a range of VSEQ frames to an oscillator. Set the BASE to the mid-point of your run of frames, and the SCALER to the number of frames in the sequence. The VSEQ oscillator effect only comes into action when the Panel is down. If the oscillator isn't running fast enough to project a new frame every frametime, the normal Trip-A-Tron pattern screen is 'plexed in allowing you to overlay patterns.
The L'FORM (Lightform) effect allows you to link an oscillator to the current pattern number. Say you had, for example, sixteen stages of a rotating pattern definition to attach an oscillator to. You'd set the BASE to 8 and the SCALER to 16, then attach the oscillator.

General Notes on Oscillator FX:
You'd de-select any unwanted oscillator FX by clicking in OFF in that effect's BASE parameters.
If 'DON'T BOTHER' is selected in the Scaler, the raw oscillator output is +/- 127.
The LOGIC section can be used to modify the output of a wave before scaling takes place. Clicking on XOR negates the wave output before it is scaled. Highlighting either the Rotate left or Rotate right icons, and selecting the number of rotations required in the adjacent Number box, will perform a bitwise shift to the left or right on the raw waveform data before scaling. The resultant value will never be allowed to exceed +/- 127, though.
Don't worry too much if you find Oscillator FX a little confusing at first. Working with waveforms and oscillators becomes second nature after a while working with the Lightsynth.
Another icon sussed, some more effects for you to play with…
Time to press on to the next section!

THE VIDEO SEQUENCER
Middle Row, Rightmost Icon
Note: VSEQ functions are only available to users with 1 Meg or more! Do not attempt to use the demo examples in this section if you don't have the memory.
To access the Video Sequencer: press either Mouse button over the VSEQ icon.

VSEQ Principles of Operation
Before I can begin to get specific about the Video Sequencer, it is necessary to understand what the VSEQ is, and generally how it can be used.
All VSEQ really is is a large area of memory, sufficient to hold a lot of picture files. By sequencing through the frames using the KML DISPLAY_FRAME command, animation will be produced. VSEQ posesses a number of options to allow you to process picture data, either individual frames or whole animation sequences. You can use the VSEQ functions to generate animations from just one source frame, if you want to.
Still, first things first, here we go…

VSEQ Naming of Parts
Examine the VSEQ screen. The dominant feature is the Frame select box filling the left-hand side of the screen. Frames available for use are either blue (an empty frame) or green (a frame with a picture in). Frames unavailable to you due to lack of memory are displayed in dark grey (unless you have a Mega ST2 or 4, this is unfortunately most of the Frame select box)! A 1-Meg system will have about 16- frames free – enough for some fun with animation at least; if you have a Mega4 you'll get over 110 frames free and you can have some Serious Fun,
Running down the right-hand edge of the screen are the eight Function icons, representing the types of image manipulation functions available to you.

Between the Frame Select box and the Function Icons you'll see boxes marked Mark Start and Mark End. These aren't buttons, they're **indicators**. More on them later. Note the Clear, Copy and Keep Palette buttons: these are used either instead of or in conjunction with the Function Icons. Then comes the **Quit** button, self-explanatory really; underneath that three buttons marked **Load/Save** (used for setting Load and Save ranges); **Source** (used for delimiting the Destination Frame or Frames): and **Dest** (used for delimiting the Destination Frame or Frames). Below that is Execute, the button which starts he image manipulation.

General VSEQ Principles

RANGES. The VSEQ can load and manipulate single frames or entire sequences of pictures, depending on what you want to do. It is easier to talk of everything in terms of **ranges**. Just think of a single frame as being a range of unit length.

There are three types of ranges in VSEQ: the LOAD/SAVE range, the SOURCE range and the DESTINATION range. You set up the ranges before you start your processing function, or before you do a SAVE.

Setting up a range is easy. Click on the button with the range-type that you require (Load/Save, Source or Dest). The MARK START indicator will illuminate. You click on the first frame of the range you're setting up. Then, the MARK END indicator will illuminate. You click on the last frame of your range, and the entire range will be highlighted – Red for a LOAD/SAVE range, Brown for the Source range, Yellow for the Destination range.

Example of setting a Load Range: Loading a frame into VSEQ memory.

Click on the LOAD/SAVE button. MARK START will illuminate. Click on the first VSEQ frame (top left of frame select box). MARK END will illuminate. As you're only loading a single frame, click on the first frame again. It should be highlighted in red. Now go to the Disk screen and LOAD the file LLAMA.RLE (this file is provided on the Trip-A-Tron disks). Once loaded, return to VSEQ.

You should see that the first frame of VSEQ has turned green, signifying that it is occupied by picture data. Use the Preview Frame option to see the piccy. (See below).

PREVIEWING FRAMES

Any frame can be *previewed* by moving the cursor over the frame and holding down the RIGHT-hand Mouse button. The display will show the frame until you release the right-hand button. You can use PREVIEW anytime, even whilst setting a range. The range ends aren't set by the right-hand mouse button. So you can use Preview to check that you're going to place the range correctly.

USING THE SOURCE AND DESTINATION RANGES

The VSEQ works by taking the frame or frames defined by the **Source Range**, somehow processing them (depending upon the functions selected), and then placing the resultant frames in the Destination Range. Many of the VSEQ functions are designed to do **in-betweening** automatically depending upon the size of the destination range. For example, you could have a Source Range of one frame, choose a ROTATE function and a destination range (say) 12 frames long. The VSEQ would spread the rotation over twelve frames in order to make a smooth animation sequence.

Example of setting and using Source and Destination Ranges

From the previous example, you now leave a single green frame in VSEQ containing the image of a llama. Using the Source Range button, mark that single frame as the Source Range in the same way as you marked the Load Range. Once you've done that press the Dest Range Button, then mark the start of the range at the second frame, and the end of the range at the top-right of the Frame Select box so that the whole top row is highlighted. Select the COPY button (it'll highlight in green) and press EXECUTE. You'll see the Dest Range frames counting down as the picture is copied to the Destination Range. When the function is done, use the Preview function to check the results.

Constraints on Source and Destination Ranges

All VSEQ functions work on the principle of ranges; you can have, say, four frames in the Source Range and twelve frames in the Destination Range (a lot more if you're using more than 1 Megabyte). Ranges can overlap; indeed the Source Range and Destination Range can be directly overlaid. The main constraint is that the **source range** must not be larger than the **destination range**.

Nearly all VSEQ functions use the **topmost frame** of VSEQ space as a working buffer, so it's a good idea not to have any vital picture info. in the topmost frame. MAP files, Sequencer use and RLE stashes all use up VSEQ memory from the top of VSEQ downwards. If you use any of these functions you'll see the blue VSEQ free frames being 'eaten up' as the memory is used up.

Using the VSEQ Functions

There are eight basic VSEQ functions available for your use. The functions are represented by the eight icons at the right-hand side of the screen. Reading top to bottom, these functions are:

Picture Reflect: Reflects the source image in either X or Y axis with or without Merge.

Picture Scroll: Scrolls the source image by a specified number of pixels up, down, left or right.

Resize and Replicate: Draws a copy of the Source image into every box defined in the Main Pattern Plot's Resize function.

Rotate: Allows rotation of the source picture in three dimensions, and transitions along X, Y or Z axes.
Filter: Scans the source image, allowing only pixels satisfying certain constraints set within the Filter to pass; also does Masking, Cutting and Intensity-Texturing.
Pixellate: Artificially degrades the resolution of the source image.
Symmetry: Allows a symmetry mode to be set, and the picture re-drawn with each pixel going through the symmetry mode. Also allows you to pass each pixel through the Trip-A-Tron main plot routine.

Cyclic Replicate: Produces multiple, resized images of the source picture arranged in a circle.
All of these functions are pretty useful if used on single frames, but when they're used over a range they're most impressive.
To select a VSEQ function for use, click on the icon representing the function you want to use. The icon will be duplicated in the **Functions Pending** area, directly below the Frame Select grid. Once an icon is here, you can either click the LEFT mouse button on it to delete it, or press the RIGHT mouse button on it to set up its options.
Example: Spinning a Llama
From the previous examples, you should now have the Source Range set to the first frame of VSEQ (frame 0), and the rest of the top row of VSEQ (frames 1-13) set up as the Destination Range. First, click on the **Copy** button if it's still highlighted, to de-select the Copy command (Copy over-rides any proper VSEQ functions in the Functions Pending area). Then, go to the **Rotate** icon and click it on. You should see a copy of the Rotate icon appear in the Functions Pending area. Go to that icon and click the RIGHT hand Mouse button.
The Frame Select grid will be replaced by the Setup options for the Rotate command. We'll go into detail about these options later; for now, just notice the wireframe oblong and the arrow buttons labelled XY, YZ and XZ. The wireframe oblong represents the screen, with a small square at upper-left for orientation. Hold down the mouse button on one of the XY arrow keys. Notice the wireframe box rotating. Rotate the picture half a turn in each plane – XY, YZ and XZ. Click on the 'P' box to see a preview of the animation. Don't make it too complex – we're only doing it over thirteen frames.
When you're happy, click QUIT, then EXECUTE. The cursor will freeze.
No, you haven't found a bug! I deliberately chose one of the more complex functions for this demo to show that stuff like 3-D rotation takes T-I-M-E! With three-plane rotation and a fairly dense image, the rotation example you just started will probably take about a minute per frame. Notice the little icon above the CLEAR button has changed from 'Make The Tea' to 'Go to Sleep'. The icon changes its message according to the anticipated complexity of the function selected (it's only a rough guide though!)
The good news is that the system isn't **totally** dead to you. Holding down the right hand Mouse Button while EXECUTE is happening will force the machine to show you what it's doing, useful for checking on the progress of a function. Holding down both Mouse buttons will abort the current function after the next frame.
When the function's finished executing, use Preview Frame on the right-hand mouse button to check the results.
Now that you've tried out a single function, I'll tell you in detail about each function and its setup screen. Click on the Rotate icon in the Functions Pending area with the LEFT Mouse button to remove it. As I describe each function, click on its icon at the right hand screen edge to bring it into the Functions Pending area, then use the **right** hand mouse button on the icon to bring up it's Setup screen. Examine the Setup screen while studying the description of the funcion's operation; Practice setting up dummy parameters for each function. When you've finished, quit from the Setup screen and click on the icon in the Functions Pending area with the **left** mouse button to remove it. Then move on to the next function.

REFLECT function:
This setup screen contains two screen-icons, divided by the X-axis in one case and the Y-axis in the other. One of the segments is highlighted in green. The picture is scanned from the highlighted area towards the axis, and the reflection is drawn simultaneously as the picture is scanned. If MERGE is selected, then black bits in the source picture are treated as transparent; if REPLACE is selected, the black pixels are treated as opaque. Click on OKAY to quit the setup zone. This function is quite fast; feel free to try it out.
SCROLL function:
Use the arrow buttons to select the number of pixels to scroll the source image. Lateral shift scrolls the picture horizontally: positive values scroll the picture to the left, negative values, to the right. Vertical shift scrolls the picture up and down: positive values scroll it up; negative values scroll it down. Selecting **WRAP** scroll type means that if the picture is scrolled off the edge of the screen, it re-appears on the opposite edge. **BLANK** scroll type means that any image scrolled off the edge if the

screen is lost. **LOCAL** scroll type means that as the source image scrolls off one side of the screen, the neighbouring frame to the source image is introduced into the space left by the source image.
Selecting PER FRAME means that the scroll values apply to each destination frame. Selecting OVER DEST RANGE means that the function will scroll each successive Destination Frame so as to arrive at the selected values by the time it reaches the last Destination Frame. For example, if you had a Dest Range of 10 frames and you were scrolling your Source by +100 per frame, all the Dest Frames would be shifted by +100; if you selected over Dest Range, the first frame would be shifted by +10, the second by +20, and so on up to the last frame shifted by +100.

RESIZE/REPLICATE function

This is largely controlled by the boxes defined in the Main Plot Routine (see the section on Main Plot Routine control functions). The only options you have are Clear Dest Frame (removes any existing image in the Dest Range before doing the function) and Merge (if Merge is selected, black pixels are treated as transparent).

3D ROTATES function

The pairs of arrows XY, XZ and YZ cause the image to rotate in the relevant plane. The Transition arrows X, Y and Z cause the image to be translated along the X, Y or Z axis. Use whatever combinations of rotation and translations takes your fancy to build up the required image rotation. You can preview the rotation at any time by clicking in the ‘P’ box.
Normally, the rotation will start from the flat, unrotated screen and proceed to whatever rotation you’ve set up. You can make it start from any position, though: just use the controls to set up the picture start orientation, click on START, then use the controls again to do the necessary. Any Previews you do after pressing START will begin with the picture in the orientation it was when you last pressed START, as will the final rotation when you Execute.
Clicking on ABS means: rotate EVERY frame in the Dest Range to the orientation set. The normal mode, OVER, means; calculate the in-between frames required to reach the selected orientation by the end of the Dest range.
Clicking on NUM replaces the wireframe picture display with four lines of numbers representing the rotation angles, so you can precisely line up rotations. Remember that in this simplified system, there are 256 ‘degrees’ to a complete rotation instead of 360. Looking at the numbers, row by row:
The top row gives the XY rotation angle, then the X-Y co-ordinate about which rotation takes place.
The next row gives the XZ rotation angle, then the X-Z co-ordinate about which the rotation occurs.
The next row gives the YZ rotation angle, then the Y-Z co-ordinate about which the rotation occurs.
The bottowm row gives the amount of translation about the X, Y and Z axes.
You can adjust the **rotation point** around which the picture spins in each plane. I recommend doing this while in Num display mode, as precise positioning of the point isn’t really possible in GRAPH mode. You use the same arrow keys as for spins and translations – switch into ROT mode to change the function of the arrow keys.
When in ROT mode, consider the arrow keys as three vertical pairs. The top-left arrows adjust the X rotation point for X-Y plane rotation; the bottom left adjust the Y-rotation point for X-Y plane rotation. The centre pair adjust the X-Z rotation point; upper set adjusting X, lower set adjusting Z. The right pair adjust the Y-Z rotation point: upper set adjusting Y, lower set adjusting Z.
Remember to switch back to ANG (normal rotate-angle mode_ after you’ve adjusted the rotation points).
I’m sorry if all this seems a little heavy – it certainly seems a lot heavier seeing it written down than it is to actually do it onscreen! Play with the function, using this text as a reference as necessary. You don’t have to EXECUTE the function to see what the results are going to look like – just use the ‘P’ preview box to see the wireframe preview. If you foul up totally, you can just leave, throw away the icon from the Functions Pending area, and get a fresh one from the right-hand column of icons. ROTATE is a powerful function. If used with plenty of memory, some spectacular effects reminiscent of the sort of stuff put out by TV effects generators are possible. Persevere – it’s worth it!

FILTER function

The FILTER function can be used to treat images, chopping out certain colours and passing others according to a variety of conditions. There’s also something weird called Intensity Texturing which is quite nice to play with,
The first option in the list is CHOP BITPLANE. You can cut out any of the four bitplanes defining a pixel – useful for, say, taking a 16-colour image and restricting it to 8-colour.
For example, many digitisers produce pictures in sixteen grey scales. Unfortunately, with only 8 levels of RGB, the Atari can only generate eight true grey scales. Often, when displaying these pictures, the colour resolution is effectively halved by having colours 0 and 1 = grey scale 0, colours 2 and 3 = grey scale 1, colours 4 and 5 = grey scale 2, etc.
Obviously, if only eight grey scales are being displayed, this is wasteful. By using CHOP BIT PLANE 1, all the lower bits of the pixels in the image are removed, forcing the grey-scale information into even-

numbered colours. You can then retain the digi pic at its original resolution whilst freeing up eight colours for your own use.

If you used CHOP BIT PLANE 4 on a 16-colour picture, it would re-draw the picture using only the lower 8 colours.

CHOP BIT PLANE is fine if you understand what bit-planes are, but you may feel more comfortable with the next function: PASS BY LOGICAL COLOUR NUMBER. Looking at the Edit screen, you'll see sixteen little green boxes. These represent the sixteen logical colours on the Atari low-res screen. If you use the mouse to turn off the boxes representing screen colours you want to cut out of the Source picture, then those colours won't be passed through the Filter.

You may want to chop out colours below, say, a certain intensity (useful for taking 'noise' off rough digi pix). For this purpose you use the RGB-intensity filter. This filter gets the RGB value of each pixel on the Source image, and compares it with the LOW threshold value (highlighted in orange on the RGB scales). If the pixel intensity is greater than that value, the pixel is not passed. To set the low pass threshold: click to highlight the SET LOW box, then click on the required RGB values to set the threshold. To set the high pass threshold: click to highlight the SET HIGH box, then click on the required RGB values to set the threshold.

Next, we have the MERGE WITH DEST mode buttons. These controls specify how the image passed through the filter is merged with any existing image in the destination frame. REPlace and MERGE modes control the behaviour of black pixels, as explained for other functions previously. MASK is interesting, and works as follows:

If the output from the filter is black, then write a black pixel to the Destination Frame; ELSE do nothing.

CUT works the opposite way about:

If the output from the filter is non-black, write a black pixel to the Destination Frame; ELSE do nothing.

I'll leave it up to you to discover the groovy things you can do with Mask and Cut. They're fun to use, especially in conjunction with digi images.

Finally, there's Intensity Texturing. This is a strange function which plots pixels of different height according to their colour. If used on tiled images, it can give a rough pseudo-3D feel to the image. It's just a weird little function which I liked, so I've left it in for you to play with. There are two types of Intensity Texturing: COL #, Colour Number texturing, where the logical colour governs the height of the generated pixels, colour 0 being the smallest and colour 15 being biggest, irrespective of the pixels' actual RGB colours; and RGB texturing, where the pixels' RGB colours are determined, the intensity governing the height; Black pixels being smallest with RGB 000, and White pixels the largest with RGB 777.

It's a weird function. Try it out.

PIXELLISE function

Pixellisation is a deliberate degrading of the resolution of the source image by rendering it in chunky pixels. It's the sort of thing you see in Question of Sport and other such quiz games where a pixellised image of some famous personality is displayed and the contestants have to guess the identity. It's quite a popular effect, so I've included it here for the sake of completeness, although I think it's a bit of a waste of time actively degrading the ST's graphics!

Choose the pixel size in which you wish the destination frame to be plotted using the arrow buttons. If KEEP SIZE ABSOLUTE is chosen, then all the destination frames will have the pixel size you choose. If you pick VARY SIZE OVER RANGE, then the image will begin at normal ST resolution and degrade gradually to reach the largest pixel size by the end of the destination range.

SYMMETRY function

The Symmetry function allows the setting of symmetry (in exactly the same format as on the SYM panels to the PLOT routine). If symmetry is set in this function, and CHOOSE QUICK SYM HERE is highlighted, then each pixel of the Source image will be subjected to the selected symmetry as it is plotted in the Destination image.

If you choose the USE INLINE EFFECTS option, then each pixel from the Source image will undergo whatever operations if the Main Plot Routine are currently active on the Main Control Panel.

The MERGE and REPLACE options alter the behaviour of black pixels, as explained for other functions.

CYCLIC REPLICATE function

This rather tidy function causes multiple images of the Source image to be produced in a circle. The higher the Order set in this function, the more images will be produced.

Examine the Cyclic Replicate edit screen,. The two sine waves at the very top of the display are the waves which generate the circle of pictures. When you first enter the cyclic Replicate screen, the waves are in the correct phase to generate a properly round circle. By clicking and holding down the mouse button on the PO- and PO+ boxes, you can alter the positions of the dots on the waves (and thereby the positions of the images on the generated circle).

Underneath the waveforms and the PH and PO buttons lie the other Cyclic Replicate controls. **You** can alter their values by using the + and – boxes at either side of the numbers.

The **Order** of the Cyclic Replicate function is the number of images to be produced in the circle. Thus a C.R. of Order 8 will consist of a ring of eight images.

The **Resize X** and **Resize Y** variables represent the required size of the images around the circle. The full Source image is scaled to fit in a box of size (Resize X, Resize Y).

The **Scale X** and **Scale Y** variables are analogous to the x- and y- diameters for the generated circle (or ellipse, if the diameters are not equal).

You can choose, for both Position of the images on the circle and Phase of the waves, whether the selected values apply equally to each destination image (ABS mode) or whether in-betweening over the destination range occurs (OVER mode).

The pattern-generation icons have now been explained. Don't worry if some of the explanations seemed a bit full of X this and Y that and Z axis rotation the other – don't worry about the maths AT ALL. I still maintain the best way to learn about VSEQ functions is by some good, dirty, honest-to-Ghod fiddling-about-to-see-what-happens. Experiment over small Destination Ranges to save time at first – as you get more confident you can commit to longer processes. The more frames you generate, the smoother will be the resulting animation.

The image-manipulation functions don't stop at just these eight single operations. The Functions Pending area can hold up to eight different icons itself – you just use strings of the basic eight functions to build more complex operations…

General Notes about VSEQ:

Loading and Saving. All SAVE operations consist of marking a frame, or range of frames, as the Save Range, then going to the Disk screen and doing a Multisave (see Disk Access section for details on this). I would recommend SAVEing all sequences and pictures as .RLE files, as they take up much less disk room than straight .NEO or .PI1 files. If a SAVE operation bombs out due to lack of disk room, you must re-enter the VSEQ to remake the Save Range.

You can LOAD into VSEQ without marking a range to load to, but it's a good habit to get into if you mark the range every time. If you can't remember how may frames are in a sequence, just mark a good deal more than you need: the LOAD routine will ignore the extra. If you **don't** mark a range, frames are loaded in are loaded above the last frame loaded, or up from frame zero if no frames have previously been loaded.

The functions **CLEAR** and **COPY** take priority over any icons in the Functions Pending queue. Remember to de-select CLEAR or COPY when you've finished using them so as not to accidentally CLEAR or COPY over something you wanted to keep!

Normally, the palette of the Source image is copied to the Destination frame(s). Sometimes, especially when using the Filter CUT and MASK options, you will want to retain the colours of the Destination frames. Select KEEP PALETTE before you EXECUTE, and the Destination Frames will retain their palettes unmolested.

Functions in a functions-pending queue are executed in left-to-right order. The result of the previous function becomes the Source Frame to the next function.

.RLE STASH MANAGEMENT

As explained in the disk section, .RLE stashes are specially-compacted animation files which can be compiled in VSEQ and animated via KML or with a stand-alone playback program. To create a .RLE stash, you must first allocate a number of frames to be used as the Stash. Mark off the frames as if they were for a standard load, then follow the instructions in the section 'Other Functions of the Disk Screen', which explains the procedure for building a .RLE stash.

Once a .RLE stash is in your VSEQ memory, either through your making it or by your loading a .BAA file, you can preview the frames in it by pressing the RIGHT-hand mouse button over any of the frames composing the stash (RLE-stash frames show up as white or pale grey). This will bring up the RLE-stash management screen, which allows you the following options:

Look at Frames. Click on this, and the current .RLE stash frame will be displayed. You can use the LEFT mouse button, or BOTH mouse buttons to run backwards and forwards through the Stash. Press the right-hand mouse button to leave the Look At Frames option.

Current Frame – You can dial up a specific frame number using these two Arrow Keys.

Delete Current Frame. Removes the current frame from the .RLE stash.

Optimise .RLE stash. If more frames have been allocated to the Stash than have actually been filled with compressed picture data, using Optimise will reclaim the unused frames for VSEQ.

Allocate Extra VSEQ: Expands the Stash by one VSEQ frame, so you can fit more compressed pictures in.

KILL: De-allocate and purge the frames set aside for the stash, Compiked .RLE stashes can be SAVEd as .BAA files, then reloaded in a single load.

That's it for the VSEQ. I could write volumes on the possible applications of VSEQ alone, especially on multimegabyte systems, but time and space will not permit. Suffice to say that some great effects are possible with VSEQ, plenty of memory and some good KML programming.

STARFIELD CONTROL: Fourth Icon from Left, Middle Row

Note The Starfield Edit screen uses up the top frame of VSEQ for a buffer. Consequently, this function is not available on 512K machines. Never fear, though – starfields can still be generated via KML programs to set uo the system variables. Read the descriptions of the startfield variables in this section.

To leave the Starfield Edit, just press ESC.

For your first attempt, I recommend that you set the Main Plot Routine up so that there's only one symmetry point enabled, and no fancy functions: It makes what you're doing to the Starfield a lot more obvious. Use the Demos to set up a simple startfield in single symmetry (for example, have the Starfield demo loaded and press 'N' to get a simple 3-D starfield, then press the right-arrow key to get single-point symmetry, then bring up the Panel and enter Starfield Edit).

Upon entering the Starfield screen, you'll see a list of the variables governing Starfield generation on the screen; superimposed upon the list of variables you'll see the Starfield itself. As you edit the variables' values, you can see instantly how the resultant starfield is affected.

Looking at the screen, you'll see listed down the left-hand side of the screen the three-letter KML names of the starfield control variables. One of the variables will be highlighted by a yellow box. Using the cursor keys (UP arrow and DOWN arrow) you can move the yellow box, highlighting each variable in turn.

Next to each variable's KML name, you'll see a line of text telling you the function of that variable.

Editing Starfield Variables

The **current variable** for editing is highlighted by the yellow box. The variable's value appears on the line at the bottom of the screen. There are two ways of altering the value of each variable: you can **move the mouse** left and right to quickly change the variable, for coarse adjustment; then you can use the **left and right arrow Cursor Keys** for fine adjustment, to get the value you want precisely.

Once a value is set up to your satisfaction, use the Cursor Keys to move the yellow box on the next variable you want to edit.

Starfield Variables – Function and Effect on Starfield Generation

From the top downwards:

SVX: Star Velocity (X) This determines how fast each star moves in the X-direction.

SVY: Star Velocity (Y). This determines vetical star speed.

SVZ: Star Velocity (Z). Stars are projected in 3-D; thus each star has **three** co-ordinates, X, Y and Z. The nearest position to the viewer is represented by z=0; the furthest position from the viewer is represented (usually) by z=4000 or so, this can be altered by settting the ZCL variable. So, to make a starfield run 'backwards' (away from the user) should require a positive SVZ. This is counter-intuitive, though, so I've set it up so that positive SVZ makes the stars zoom **towards** you, and the negative SVZ makes the stars zoom **away** from you. Mess with the variable and see.

ANG: Star Angualr Velocity. The stars can be made to spin around the current star rotation centre point by assigning to them an **angular velocity**.

ZCL: Z Clipping Plane. Determines the maximum distance from the viewer at which stars are still visible. Stars travelling beyond ZCL are shut down.

ROX: Rotation Point (x)

ROY: Rotation Point (y) Between them these two co-ordinates determine the point about which the stars spin due to any angular velocity or Absolute Image Angle settings.

PRO: Projection Depth. The setting of this variable determines the degree of perspective shift given to stars as they get further away from the user. Generally, the closer to zero this variable is set, the more extreme the perspective shift.

ABS: Absolute Image Angle. The entire Starfield image can be rotated by a given number of Yakly degrees around the rotation point by the setting of this variable.

INX: Star Initialise Point (x)

INY: Star Initialise Point (y)

INZ: Star Initialise Point (z)

These three variables determine where the next star is to be generated in 3-D space. Setting any of these variables to –1 means 'generate randomly everywhere along the axis'. For example, 'horizon' effects can be generated by setting INZ=2000, SVZ=50, SVY=0, SVX=0, INX=1 (anywhere along the x-axis), INY=140.

INR: Star Initialise Rate. This determines the rate at which stars are generated, with 1 being the fastest rate. Setting this variable to –1 shuts down star initialisation, allowing the creation of special effects by manually forcing star initialisation with the KML ISTAR command.

SCL: Star Colour (Low). The colour in which stars are first plotted when they're generated.

SCH: Star Colour (High). The maximum colour number which stars can attain before either returning to the low colour or reversing colour transition (see below).

SCM: Star Colour Transition Mode. Once generated, the colour of a star changes from the SCL setting to the SCH setting at a rate governed by the SCM variable. If SCM is zero, once SCH is reached, the star colour 'wraps around' to SCL. If SCM is 2, the star colour 'bounces back' to SCL and continues to oscillate between the two values.

SCR: Star Colour Transition Rate. The speed at which the star colour changes. 1=fastest.
TRA: Star Trailback Length. Each star can be shown either as a single dot, or as a streak of length TRA pixels. Maximum Trailback length is 7.
Use the mouse and keys to mess around with the variables, observing the change in the generated starfield. You'll soon get the idea of what the variables can do. Amazing Starfield effects are possible under KML, as you alter the variables via the mouse, oscillator or whichever way you like, continuously, as the 'field is generating.

Getting Starfields on the Main Display

Starfields created in the Editor will not be displayed until turned on by a KML STARF 1 command. If you want more than one Starfield assigned to the keyboard, you must note down the settings in the Starfield Editor and translate them into a KML program. For example, the following KML code fragment will generate a horizon-effect starfield:

```
STARF 0
WAIT 8; Turns off any existing starfield and allows it to clear
SET 0, SVX
SET 0, SVY
SET 0, ANG
SET 0, ABS
SET 50, SVZ; Set up star velocities
SET 2000, INZ
SET –1, INX
SET 140, INY; Set up star init parameters
SET 1, INR; Maximum star density
STARF 1; Turn on the stars
```

If you added the following, you would be able to tilt the horizon by moving the mouse:

```
LOOP
SET CX, ABS
WAIT 1
FOREVER
```

Limitations on Using Starfields

Because starfield generation is a lot more complex than the simple '2-D pixels moving at different rates pretending to be 3-D' approach used in Colourspace 1, mixing Starfields and Pattern effects is possible, but only if very thin, uncomplicated Starfields are used. This is due to **interrupt overrun**. Pattern generation takes place in foreground time, starfields on interrupts; if the interrupt routine takes all the time up, there's no time for Pattern effects.
This isn't such a big handicap when you consider that Starfields are now a performance effect in their own right, rather than just pretty backgrounds. Using KML you can place any of the parameters for starfield generation under mouse control. You can mix laser effects with starfield, too, so you won't lack for pretty patterns.
If you want REAL power, contact us with a view to getting a video mixer, I use a custom-built five-channel mixer; with such a set up, and multiple Ataris linked via MIDI, you can mix starfields, lasers, animations, pattern generation, everything onto one display in up to 80 colours!

THE SILLYSCOPE: Fifth Icon from Left, Top Row.

This is another of those functions which is sure to look hopelessly complicated written down on paper than it actually is to use. I'll explain all the functions and how they work, but don't worry if it all sounds heavy. Jump in there, play with the controls, and you'll soon get the idea.
Before you go into the Sillyscope screen, ensure that you've got some waveforms loaded. Set both Post- and Pre-Symmetry to a single point, just so that the Sillyscope screen responds faster and your changes to the parameters more obvious.
Enter the Sillyscope screen by pressing either Mouse Button over the Sillyscope icon.

The Sillyscope Edit Screen – Taking Stock

Examine the screen. You'll notice that the screen is dominated by a large blank area where the 'scope displays actually appear during editing. If either of the 'scope displays are on when you enter the screen, you'll see them generating here.
Below the display area lies the 'scope control panel'. At the left you'll see a cluster of four buttons: X-1, Y-1, X-2 and Y-2. These represent the X and Y beams of laser generators 1 and 2. The current beam being edited is highlighted in green.
Next to these, notice two coloured buttons. These are the buttons which switch the laser generators ON and OFF. The top button is for laser 1, the bottom one is for laser 2.
Each laser can generate two types of display – Dot and Line displays. In DOT mode, the laser trace consists of thirty-two dots. In LINE mode, the dots are joined by coloured lines.

The coloured buttons are BLUE if the traces are OFF. Clicking on a blue button with the LEFT hand Mouse button turns that trace ON in DOT mode – the button will turn GREEN. Click on a blue button with the RIGHT mouse button to turn the trace on in LINE mode – the button will turn RED.

Try doing this a bit. Turn on and off the traces in both Dot and Line mode to see what happens. When you've finished, leave one trace turned on so that you can see what occurs as you edit.

While learning, it's probably best to edit in DOT mode. Dot mode is faster and more forgiving – LINE mode traces can look untidy unless they're smoothly constructed.

The laser generators rely totally upon waveforms for their operations. Each laser trace has two **Master Waveforms**, the X and Y waveforms, which between them determine the way in which the laser scans. Look at the 'scope panel, and notice the central box with a waveform in it. The nine buttons underneath determine which of the eight waveforms currently loaded is being used by the function being edited. Immediately to the right of the waveform display you'll see three buttons labelled SWEEP, PHASE and AMPLI. More on these later. To the right of those is a cluster of four buttons – XCENT, YCENT, TRAIL and COLR – and two Numeric boxes.

Altering the Master Waveforms

Altering the Master Waveform of either trace is easy. Ensure that none of the buttons to the right of the waveform display is highlighted, select the trace whose Master Waveform you want to alter – X-1, Y-1, X-2 or Y-2. The Master Waveform will appear in the waveform display. To change it, click on any of the eight Waveform Buttons underneath the display. If the laser generator is turned on, you'll see the generated pattern change accordingly.

Sweep, Phase and Amplitude Controls

These controls are unique to each individual Master Waveform, and determine how the waveform is scanned and fed to the Laser Generator.

SWEEP: This is the rate at which the Master Waveform is read out. A waveform table has 256 entries. The Sweep value determines the spacing between the 32 points comprising the laser trace. A Sweep value of 8 would mean that one complete cycle of waveform would be read out each time the display is drawn.

PHASE: This is the amount by which the Master Waveform is phase-shifted. Altering this will cause the pattern shape to change, flattening or expanding, depending on the nature of the waves you've got selected.

AMPLI: This is the maximum amplitude for the Master Waveform. Hooking the amplitude of the Master Waveform to another oscillator can be interesting.

ADJUSTING THE VALUES:

The procedure for adjusting the values of Sweep, Phase and Amplitude is as follows:

Highlight the box of the function whose value you want to adjust. Each of these functions can be either a constant or the output of yet another waveform. When you highlight the function you want to edit, look at the Waveform Display. If it shows a wave, then the value is linked to an oscillator. If no wave is present, the value is a constant.

EDITING A CONSTANT

The current value of the constant will appear in the lower Numeric Field at the right side of the screen. You can adjust the value of the constant by clicking on the numbers with left and right-hand mouse buttons to add and subtract, as explained elsewhere.

EDITING A WAVEFORM-LINKED VALUE

You can change the waveform to which the value is linked by pressing any of the eight Waveform Select buttons underneath the waveform display. The value produced by the linked oscillator is determined also by the two Numeric Fields, the top Field represents the wave speed of the controlling oscillator. The bottom Field represents the amplitude of the resultant output. Edit the two Fields with the mouse, as previously described.

CHANGING FROM WAVEFORM-LINKED TO CONSTANT

If a variable is wave-form linked and you want to make it a Constant, click on the right-hand, larger button underneath the Waveform Display. The displayed waveform will disappear and you can go edit the Constant in the lower numeric field. Likewise, you can turn a constant into a waveform-linked variable by just clicking on one of the Waveform buttons, and adjusting the wave speed and amplitude accordingly.

Xcent, Ycent, Trail and Colr variables:

These variables differ from Sweep, Phase and Ampli in that they apply to the Laser Generator overall, rather than the two separate Master Waveforms. Thus, the XCENT would be the same for both X-1 and Y-1. The variables can be edited and attached to waveforms in exactly the same way as Sweep, Phase and Ampli.

XCENT: Determines the horizontal positioning of the laser trace.

YCENT: Determines the vertical positioning of the laser trace.

TRAIL: Amount of trailback per trace, up to a maximum of 31.

COLR: Colour of each dot or line in the laser form.

Maximum and Mimimum Values for the functions:
All these functions have certain limits beyond which they cannot be set. These limits are as follows:

	CONSTANT	**WAVE AMPLITUDE**
SWEEP	+/- 128	128
PHASE	+/- 255	255
AMPLI	+/- 400	400
XCENT	+/- 500	500
YCENT	+/- 500	500
TRAIL	0-31	15
COLR	0-15	8

Seeing your ‘Scope Traces on the main screen:
If you leave SillyScope with traces enabled, they will appear as soon as you lower the main Control Panel. You can turn traces on and off either by going into Sillyscope or by using the KML SCOPE command. If you want to assign traces to keys in KML, you need to use a special set of KML commands which allow you to define the control variables iin terms of waveforms and amplitudes – see the KML section on the subject. Under KML, all parameters of ‘scope pattern generation can be placed under mouse control if necessary, allowing great flexibility of display.

Limitations of SillyScope Displays
Like starfields, SillyScope displays produce interrupt over-run and so you can’d do standard Trip-A-Tron pattern generation while they’re on. You can display Starfields and Sillyscope stuff together though.
Dot-mode displays are subject to all the Trip-A-Tron main plot routine settings. Line Mode displays, however, use ST’s Line A linedraw routines, and so ignore the Trip-A-Tron main plot routine. I’ve set up the Line Mode displays so that they can pic up the Presym settings; even if Presym itself is de-selected, provided that some symmetry points are defined on its Presym panel, the Line Mode laser traces will pick this up and use it.

General Notes about using SillyScope:
When you’re learning, use Dot Mode and don’t have anything too fast-moving. Keep the Wave Speeds of oscillator-linked variables low. Use simple waveforms which aren’t too jagged – straight Sine waves are ideal. Use Line Mode sparingly until you understand what you’re doing. Above all, don’t be daunted by the complexity of what’s going on – experiment, learn and have a zarjaz time.

GLOBAL PARAMETER SETUP: Fourth Icon from Left, Top Row
The GLOBAL PARAMETER screen is used to set certain system options affecting the cursors and plot speed.
To access the Global Parameter Setup Screen, click on the Global Parameters icon with either Mouse button.
The functions of the Global Parameters screen are accessed by a bunch of sliders and buttons. I’ll run through them in turn:
Buffer Length. This slider determines the maximum number of pattern generation points the system can keep track of at one time. The longer the Buffer Length, the larger the pattern trails generated by Trip-A-Tron can be. Obviously, with longer pattern trails there’ll be correspondingly more plotting for the stystem to do, so pattern inertia will increase. Experiment with the value and notice its effect, especially when you’ve got a lot of Symmetry going on.
Secondary Point Trailback Length. This slider determines the distance at which the secondary cursor follows the primary cursor when enabled. Short Trailback Length settings and Line Mode patterns can be quite interesting.
Smoothing Delay Value. Every pattern drawn by the system persists after it’s been drawn for a time dependant on the setting of this Slider. If you take this right down to zero, you’ll notice that the pattern flow becomes less smooth as the limiting factor becomes the actual plot speed of the patterns being drawn – large patterns will slow the system down when trying to do fast traces. Adjust the Smoothing Delay value according to taste; the ideal setting will depend on the size and type of pattern you’re drawing and the speed of performance – fast and zappy, or slow and flowing.
Screen Mode. You can set the unusual **interlace** mode at this control. Interlace is a cheap way of getting x-axis reflection; odd-numbered lines are inverted and interlaced with even-numbered lines to give the illusion of twice as many pixels on the screen. Try it – it can look OK under the right circumstances, but you certainly won’t want it all the time. INTERLACE doesn’t take effect until the main Control Panel is lowered.
ONTIME and OFFTIME. The settings of these two sliders determines how pattern generation occurs when the mouse button is held down. If the OFFTIME setting is zero, then regardless of ONTIME, one pattern will be started at the cursor position every frametime. If OFFTIME is non-zero, then;
For a count of ONTIME consecutive frames, a pattern is generated every frametime; then
For a count of OFFTIME consecutive frames, nothing happens. Then the cycle repeats. For example, if ONTIME=1 and OFFTIME=1 then a pattern is generated only every two frametimes.

Mouse Cursor Mode. In WRAP mode, moving the cursor off the edge of the screen will cause it to re-appear on the opposite edge. In CONSTRAINED mode, the cursor stops when it reaches a screen edge. Constrained mode is useful if you're working with the cursor visibly turned off – it stops you 'losing' yourself so easily!

Mouse Button 2 Mode. This determines the function of the second Mouse button. In VECTOR mode, moving the mouse and, while it's moving, holding down button 2, locks the cursor onto a vector equal to the velocity of the mouse at the time you pressed the button – useful for doing dead-straight lines in a performance. In CENTRE mode, mouse button 2 returns the cursor to the position (160,100) – useful for locating yourself if you're working with cursor visibly off and you get lost. In KML #128 mode, pressing the right Mouse Button will cause the execution of any program currently residing in KML event 128.

VAMOS. Spanish for QUIT.

Global Parameter Bank. You can assign up to sixteen lots of different Global Parameter settings, and switch between them either by clicking on these buttons or via the KML GLOBAL command. Global Parameter settings also include the current settings of the Resize Boxes for the main plot routine. All sixteen lots of Global Settings can be SAVEd and LOADED as a .PRE file.

MIDI TEST/SETUP SCREEN: Top Row, Second Icon from Left

This screen allows you to check the MIDI connections between linked ATARIs, or between an external synth and the ATARI. To enter the Midi Test screen, click either button on the Midi icon. Upon entry to the screen, you'll notice two windows: the left-hand window contains the **outgoing** test wave being sent over MIDI, the right-hand window shows anything coming in over MIDI.

In linking multiple TRIP-A-TRONS together, you must connect them serially using MIDI cables and assign each a Node Number. (See the KML MIDI section for the full details). Assuming you were linking two machines, you'd use this screen to set one to Node One and the other to Node Two (the vertical row of boxes labelled 1-8 on the left). If you looked at this screen on the second Atari, Node Two, you'd see the signal from Node One appearing in the RECEIVE box, thus confirming that the MIDI link between nodes one and two was OK. You can hook up to eight nodes into the system, and check them out using this screen. You choose which node to send your test wave to by clicking on the correct node number in the SENDING TO NODE column.

If you've got a MIDI musical instrument hooked up to the Atari MIDI IN line, then you can check that it's sending data correctly by clicking on SYNTH. When in SYNTH mode, pressing keys on the external synth should cause the line in the RECEIVE box to move about accordingly.

When you're done testing the MIDI connections, leave the screen by pressing NAFF OFF.

THE SEQUENCER (Top Row, Third from Left)

To enter the Sequencer, press either Mouse button on the Sequencer icon. If a sequence has not previously been loaded from disk, and this is the first Sequencer access of a session, a requester will appear asking you to OK the sequencer taking a frame of VSEQ for storage. Provided you allow the Sequencer to do this, the control screen will fade in.

NOTE The Sequencer takes up one frame of VSEQ memory to store the sequences. For this reason the Sequencer cannot be used on 512K systems, and access to it will not be allowed on such systems.

Sequencer Overview

The Sequencer built into Trip-A-Tron is a simple program allowing the entry of event data in realtime on eight channels, and subsequent playback with event triggering at the correct time during a performance. Edit commands allow the cutting and pasting between tracks of sequences or bits of sequences.

Examine the Sequencer screen. It's quite complex, but the controls are spread into distinct blocks on the screen and the function of each block is easy to understand. Here's a brief description of each control block area and its function.

EVENT ENTRY SETTINGS: At the upper left, this block contains nine columns of tiny boxes, with the figures 1-9 along the top and 1-8 down the left hand side. This is where you set up the events you require to be entered into the sequence when you press the Event Entry keys.

SEQUENCER RUN CONTROLS: Four boxes at top-centre, with RUN, COUNT, KEY and IMM written in them. These controls start up the sequencer and determine whether the sequencer does a countdown before it begins to run, waits for a keypress, or runs straight away.

RANGE MARKER STORES: Top of screen, just right of centre; eight boxes with A-H insie them. Each of these markers can contain the start and end-points of specified sequencer segments during editing.

CURRENT POSITION CONTROLS: Directly below the Range Marker boxes, these controls consist of a set of four arrow buttons, three boxes and time display. The time display shows the current position in the sequence, and the arrow buttons may be used to move around within the sequence.

ZOOM CONTROLS: Upper-right of the screen, just below and to the right of the camel's head, two boxes marked OUT and IN. These buttons allow you to view the sequence display in a variety of scales.

TRACK ALLOCATION BUTTONS: Right-hand side of screen, underneath the Zoom controls, eight buttons labelled DISPLAY TRAX. You can de-allocate unused tracks with these buttons. If you're only recording on four tracks, de-allocate the unused four before the session. The memory freed by de-allocating tracks is given to active tracks on demand.

SEQUENCE DISPLAY: Horizontal band across the screen, almost halfway down. The sequences you enter can be seen in graphical form in this scrolling window.

EVENT NUMBER BOX: Left of screen, just below centre, a box with EVENT # and FORCE inscribed within. This box is used in conjunction with Event Entry controls, in order to denote which KML event number is added to the Sequence; or with the Sequence Display, to change the event number of an event already in the Sequence.

EDIT CONTROLS: Large box containing many smaller ones, right two-thirds of the screen, just below centre, topped by seated, opposing llamas, some grass, two mountains and a rainbow. These controls are used to set up the Edit functions, allowing the movement of sequences from track to track, or the merging of one track with another.

PULSE DISPLAYS: Eight large boxes at bottom of screen. These displays pulse according to tracks 1-8 of the Sequencer during playback, partly because Yak thought it'd look groovy, and partly because it can be a help to see previous tracks playing back while you're recording a new track.

QUIT THINGY: The Thingy (well, it's not exactly a box) lies at the bottom right of the screen. You can leave the Sequencer by clicking on it.

Now you know where all the buttons are, I'll proceed to explain the use of the Sequencer and the function of all those controls.

Recording a Sequence

Sequence data is entered, in real-time, while the Sequencer is running, by using the Event Entry keys. These keys are the **numeric-pad keys 1-9**. You use the Event Entry controls to determine which events will be entered into the sequence by these keys.

Examine the Event Entry box at upper-left. You'll notice a grid of boxes, with the numbers 1-9 at the top and 1-8 down the side.

The top row of numbers 1-9 represent the Event Entry keys. The column of coloured boxes beneath each number represent the events and track-number of events to be entered when you press the corresponding key during data entry.

For each key, one event on tracks 1-8 can be entered. Examining the column of boxes under each number, DARK BLUE boxes mean that no event is to be entered on that channel by that key, and COLOURED boxes mean that an event WILL be entered. If you look at the default settings, you'll see that event entry key #1 is set to enter a single event on track 1, and key 9 is set to enter an event on all eight tracks.

To alter the settings, just click on an empty box to turn it on, or an existing box to turn it off, using the LEFT Mouse button.

Fine, you can tell the system that if you press Event Entry key 1 it will enter events on any channel, but which events?

To set the actual event numbers, you first assign the channels as previously explained. Then, click on the assigned box with the RIGHT-hand Mouse button.

The Event Number Box will highlight in red, displaying the event number and Force value. By clicking on the numbers inside the box, you can change these values to those you require. Click anywhere outside the box to enter the values when you're done – the Event Number Box will un-highlight, and your values will be assigned.

Set the event-numbers for any other tracks you've got assigned in the same way.

NOTE about Event Numbers and Force Numbers: The Event Numbers correspond to KML event numbers. During sequencer playback, if event 5 is encountered on track 4, say, then KML program 5 will be run on parallel channel #4. Force numbers are an optional parameter which is passed to the KML system and can be read by a KML command. They're similar to MIDI velocity codes, but since the Atari ST keyboard is not velocity-sensing, you can set the Force along with the Event Number. For example, you could write a KML strobe routine which checked for Force value and performed a correspondingly brighter strobe for higher values. Using different Force numbers would then allow you to get a variety of strobe intensities while only using up one Event slot. If you don't need that sort of facility, just ignore the Force number.

Having set up the Event Entry keys to provide the correct events on the channels you want, the next step before you start recording is to set into RECORD mode the necessary channels.

Look again at the Event Entry controls. Notice that the channel numbers, 1-8 down the left-hand side, each have a small green light to the left of them. These lights are the **sequencer record buttons.**

A green light by a track number means that the track in **Play** mode, and no events will be placed on that track by pressing the Event Entry keys. A red light by a track number means that the track is in **Record** mode, and that events will be recorded by pressing the Event Entry keys.
You can change the individual tracks rec/play status by hyst clicking on these little buttons. Before you record, set the tracks you're going onto to RECORD mode using these buttons.
To actually record your sequence, you use the Sequencer Run controls. When you click in the RUN button, you'll start the sequencer in a variety of ways, according to the setting of the start mode. The default start mode is COUNT; you can change the mode to IMM or KEY by clicking on the appropriate button before pressing RUN.
In COUNT mode, the sequencer will strobe off five sections countdown before actually starting the sequencer. This gives you time to get ready.
In KEY mode, the sequencer waits and then starts up automatically when an Event Entry key is pressed.
In IMM mode, the sequencer starts the moment you press the RUN button.
Once the sequencer has started, you tap the relevant Event Entry keys at the required times in order to record the event numbers into the sequence. As you do so you'll notice the pulse meters at the bottom of the screen moving accordingly. Once you've recorded your sequence, press ESC on the keyboard to stop the sequencer and return to edit mode.
NOTES about running the sequencer:
When you click on RUN with the LEFT mouse button, the Sequencer runs from time 00:00.0, and any events existing on tracks placed in RECORD mode are deleted, ready for the new data to be recorded.
If you click on RUN with the RIGHT mouse button, the Sequencer runs from its current position, and Record tracks are only cleared from the current position onwards. Material before the current position is retained.
If you're recording multiple tracks one after the other, or you're just playing back a sequence you just recorded, REMEMBER to switch the track(s) you just did back into PLAY mode before RUNning the sequencer again, otherwise you'll lose your data!
Try This: Using the knowledge so far gained, lay down an arbitary sequence, switch all tracks to PLAY and run the sequencer to watch the playback. You'll notice all the pulse meters responding as the events you've recorded pass through the sequencer.
The arrows work on three scales. In the lowest scale, clicking a single arrow moves the sequence by 1/50 sec, and a double arrow by 1/5 sec. In the medium scale, a single arrow moves by 1 sec, a double arrow by 10 sec. In the high scale, the single arrow moves the sequence by 1 minute, and the double by 10!

The scale method of the arrows is set by the three buttons just below the arrows. Click on the rightmost button to set **Low** scaling, the centre button for **Medium** scaling, and the leftmost for **High** scaling.

So, using the arrows and the scaling buttons, you can move to any time within 1/50 sec., within the sequence. There's another, move intuitive way to move aroun d the sequence; use the Sequence Display.

As you move the current point, you'll notice a graphical representation of the sequence data displayed in the Sequence Display. Each blip on the display represents an event on a channel in the Sequencer. Channel 1 is at the top, channel 8 at the bottom. Events to the left of the Current Point are in the future; the Current Point, between the white dots, is Now. Clicking on any of the time in the past or future moves the Current Point so that the selected time becomes Now. The display scrolls accordingly. You can travel anywhere in the timespace of the sequence by this method.
Changing the Scale of the Display
You can alter the timescale of the graphical representation by using the two ZOOM buttons, IN and OUT. Zooming IN decreases the amount of sequence-time displayed around the current point, meaning that data 'blips' get bigger, allowing for more precise editing. Zooming OUT means that more time is displayed around the current point, the blips are correspondingly squished-up but you can get a much more general overview of a sequence. Eight levels of ZOOM are available.
Editing a Sequence in Memory
Changing an Event:

You can change the Event and Force numbers of any event in the sequence. To do so, position the event's blip between the two Current Position markers. Use ZOOM if necessary to ensure that the blip is exactly on the current position line. Click on the event's blip. Notice two horizontal lines running through the display – they show which track is currently being examined. The lines bracket the selected track. When you click on an event's blip, ensure that the current track markers bracket the blip.

Once the blip has been clicked on, go to the Event Number box, and click anywhere inside it. The Box will highlight and the blip's Event Number and Force Number appear inside. The numbers can be changed by clicking on them in the usual manner; anywhere outside the box to accept the altered values and return to edit mode.

Inserting an Event at the Current Point:

Move the CP to the place where you want to insert your event(s). Then, set-up one of the Event Entry keys exactly as if you were going to run the sequencer. When you're done, switch the relevant tracks to RECORD and click **on the Event Entry Key number**, above the columns of boxes. The event(s) will be entered at the current point exactly as if you'd pressed the Event Entry key during recording.

Deleting an Event at the Current Point:

Get the unwanted event's blip at the Current Point, its track bracketed by the Current Track Markers. Go into the Edit Controls box, and click on DELETE. (The one at lower left og the Edit Box. The other function is DELETE-TIME, which is different). It will highlight. Then, click anywhere outside the Edit Controls box to perform the DELETE. When you're finished click on DELETE again to un-highlight and deselect it.

Deleting a Track:

Select the track to delete by positioning the Current Track Markers on it. Then, go to DELETE, highlight it, and highlight also TRACK, next to it. Click anywhere outside the box to delete the current box. Click on DELETE again to deselect the function.

Deleting Time on a Track:

You can insert a specific amount of time at any point on any track. Events to the right of the current point are shifted rightwards (forwards in Time) accordingly. To do this, move the Current Position to the point where you wish to insert some time. Place the current track markers on the track you're inserting on.

Go to the Edit box, click on INSERT (TIME) – the lower-right in the Edit box – you will see a figure appear below the TIME notation. This figure represents, in 1/50ths of a second, how much time to insert. You can click on this number to edit it, up to a maximum of 99 (i.e 2 secondsworth). Now, each click outside the Edit box will insert that amount of time into the selected track. Continue adding time until you've got enough, then click on INSERT (TIME) to deselect the function.

Deleting Time on a Track:

Proceed as for inserting time, except that you highlight DELETE (TIME) in the edit box instead of INSERT (TIME). Events to the right of the Current Point (in the future) are shifted to the left (back in time) towards the Current Point (now). Events shifting beyond the Current Point due to a Delete Time command are discarded (taken out of Time).

RANGES – Using Ranges on Sequencer Edit Mode
Some functions, such as MERGE and COPY between tracks, demand that a range containing sequence data be set. This next section describes how to make a range, and what you can do with it.

Defining a Range:

For the purposes of this sequencer, a range is defined as a portion of sequencer event data, one one track, between a start-point and an end-point. You define a range in the following manner:

Position the Current Point at the start of your range, and ensure that the Current Track markers are bracketing the track your data is on. Pick an empty (unhighlighted) Range Marker button and click on it. It will go green, meaning that it's ready to accept range data. Click on the button again to make it the Current Range (the button will go red). Then, click on MARK START in the Edit box to denote the start of the range. A red blip will appear in the display, marking the start point of the range.

To mark the endpoint of a range, move the current point to the end of your range and press MARK END. A second red blip will appear, denoting the end of your range.

NOTE about the Range Markers: You can store up to eight range definitions in the Range Markers. A marker with no data attached to it is unhighlighted; a marker with range info set up is highlighted in green; the marker representing the current range is highlighted in red; MARK START and MARK END only affect the current range; if none is selected, they have no effect. Likewise, the Range-based edit commands all work only with the Current Range.

To make any range the Current Range, cluck on its green Range Marker with the LEFT Mouse button to select it. To deselect the Current Range, click on its red range marker with the LEFT Mouse button. To clear a range marker, either re-define its start and endpoints, or select it as the Current Range, then cluck the RIGHT Mouse button on its red range marker. The marker will be cleared.

With a range demacated and selected as the Current Range, the following extra commands are available.

Merge at Current Point:

Place the Current Point at the place and track where you wish to insert the data defined in the Current Range. Select MERGE, then click outside the box. The range data will be merged with the current track at the current point, over-writing ant events already there. Click on MERGE again to terminate the function.

Cut at Current Point:

Proceed as for MERGE but select CUT instead. Clicking outside the box will, for each blip in the current range, delete an event on the destination track, beginning at the current point.

MERGE-EVERY and CUT-EVERY

These are special cases of the Merge and Cut commands. They work by scanning the destination track for a specific event, and then MERGEing or CUTting the source range data at that point; then, they look for the next occurrence of the event, and repeat the process there, for the duration of the track.

Using this technique you could, for example, replace every event #1 on a track with another event or sequence of events.

To perform MERGE-EVERY and CUT-EVERY, position the current point at the position from which you want the search to begin. Highlight MERGE (or CUT if you're cutting), then highlight EVERY. A number will appear underneath EVERY; this is the number of the event to look for on the destination track. Edit the number to that required by clicking on it. Clicking outside the box will search for each occurrence of that event number on the destination track, and at each occurrence a MERGE or CUT will be performed.

When done, de-select the Merge or Cut operation by clicking on its button.

DELETING A RANGE:

With a range set and current, the DELETE function will delete the data between the range markers when activated.

MOVING A RANGE THROUGH TIME:

With a Current Range selected, the function of the insert (TIME) and Delete (TIME) is altered. What happens is that, for the selected time value, the **whole range** is shifted to the left (if Delete Time is selected) or right (if Insert Time is selected). Data displaced by the moving range is overwritten so watch it.

Other Sequencer Controls

DISPLAY TRAX buttons

These buttons determine which tracks are active (and therefore available for use). Unwanted tracks can be turned off by de-selecting the relevant button (leftmost button represents Track 1, the right most Track 8).

Tracks can be turned ON or OFF at will, or completely de-allocated. To de-allocate a track (and thereby free up its memory), first turn it OFF, then click on the deselected box with the RIGHT Mouse button, its box will highlight in white, representing a de-allocated track. If any of the other tracks runs out of memory, it will look for a de-allocated track and grab it.

To regain a de-allocated track, click the RIGHT button on its white box to clear it, then the LEFT button to re-allocate the track. Note that if you have RUN the sequencer in the meantime, you will lose any data previously on the track.

Notes About The Sequencer

Memory. Each track can hold up to 664 events. This can be extended by de-selecting unused tracks, any track going over 664 events will pinch the memory from an unallocated track, thereby allowing itself another 664 events.

Should a track run out of memory with no more de-allocated tracks available, the sequencer will drop out of RECORD mode automatically.

Marks. If not end marker is set, the system defaults to the end of the marked track.

Shortcuts. To return the display to the beginning of the sequence, click on the **left-hand llama**. You can jump to the start-point of any range marked (**except** a range marked in red, the Current Range) by clicking the right-hand Mouse Button on its Range Marker.

PUTTING THE SEQUENCE TO WORK

Before you can use a sequence constructed in the Sequencer, you must make sure you have the right KML events in memory for the sequencer to use. To start the sequencer running, you must use the KML command RUN-SEQUENCER. Once this command has been issued, the Sequencer will start running and continue either until the Panel is raised or a STOP-SEQUENCER command is executed.

The Sequencer starts running at the Current Point; to start it at any time in the sequence you use the command SEQ-TIME, specifying the time you want the sequencer to start from, followed by the usual RUN-SEQUENCER command.

INTERRUPT-OVERRUN plagues us again

Because the Sequencer relies on 1/50 sec. interrupts in order to maintain correct timing, beware of interrupt-overrun caused by the use of Starfield or Sillyscope displays, or by too-long KML programs with few WAIT statements. If you load the system so that interrupt-overrun occurrs, **sequencer timing will be knackered.**

This is a drag, but I'd recommend that for really precise timing of events like starfields and 'scope displays, you use an external machine triggering events remotely via MIDI. In any multi-Atari setup, always have a system for the sequencer which doesn't have to do intensive stuff like starfield. With judcious use of EXTERN commands, one sequencer can sequence events remotely on up to seven other machines.

A Hitch-Hiker's Guide to KML

Whereas Trip-A-Tron can be adequately used by just setting-up the parameters from the various edit screens available, doing this during a performance is not really possible. A control method for changing system parameters without needing to use the Control Panel is needed, and that's where KML comes in.

On the simplest level, you can use KML to assign commands to the keys of the keyboard. Any key can be made to execute any command, or sequence of commands, that you require.

KML stands for Key Macro Language. KML is in fact a simple programming language, with variables, conditional branching, and all the other stuff which goes with programming. However, even if you're not a programmer, don't worry, because KML is very simple, easier even than BASIC.

KML also has some unique features. You can have up to 128 KML programs in memory at once, and more importantly, you can run up to 8 programs at once.

Now, let's have a look at the fundamentals of KML. You need to know how to use the Editor and Key Assign Screens.

Fundamentals: The KML Editor

Enter the Editor by pressing the RIGHT Mouse button over the KML icon (fifth from left, middle row, with KML writ large upon it). The KML screen will appear, and, if you've got any demos loaded, it'll probably have a KML program already in it.

Looking at the edit screen, you'll see that it is divided into two halves. The top half is the **program area**, where the current KML program appears as a listing. The bottom half gives the current event number, the remaining free RAM, and the version number, and perhaps could have been smaller, but that's the way Yak programmed it, so there it is.

When you enter the editor, the cursor (an underline character) will be on the top line, at the left-hand side. You can move the cursor up and down through a program listing by using the arrow keys on the cursor cluster; it can be moved from side to side, along a program line, using left- and right-arrow keys. If you cursor off the bottom of a listing which is longer than the screen, the screen scrolls to reveal the next line.

KML program commands can be entered on any empty program line. You can get to an empty line by cursoring to the bottom of a listing, going to an empty program area, or inserting a fresh line between two existing lines.

Type as normal to enter program text, and press RETURN to enter your line. If you have made an error typing the command, the screen will flash red, and KML will not accept the line. Re-edit the line to correct the error using the cursor and Backspace keys (the Backspace key zaps the char to the immediate left of the cursor).

To insert a new empty line between two existing lines, place the cursor at the beginning (left-hand-end) of the line **before** the one in which you want your code inserted. Press RETURN, and a new line will open up with the cursor in it ready to accept your new line.

To delete an existing line, place the cursor at the beginning of the line you wish to zap, and press Backspace, the line will be zapped and the program code closed up.

To o between adjacent program areas, **hold down SHIFT** and press the right-arrow key (to go to the **next** program area) or the left-arrow key (to go to the **previous** program area).

To leave the Editor and return to the main Control Panel, press ESC.

Before you try anything in KML, be sure to read the section 'about KML' and a good look through the available commands.

The Key Assign Screen:

To gain this screen, press the LEFT Mouse button over the KML icon. This will yield a picture of an ST keyboard, and if you have any demos loaded you'll probably see that some of the keys are highlighted in red. These are keys which already have programs assigned to them. If you pass the mouse cursor over these highlighted keys, you'll see each key's assigned program number and channel number appear in the EVENT # and PARALLEL CHANNEL boxes.

Assigning a key is simple. You just click the mouse on an un-assigned key. The key will highlight in red, and the number ‘1’ will appear in the EVENT # box. Move the cursor to the Event # box, and by clicking on the number there in the usual manner, you change the Event number to that of the KML event you wish to assign to the key. If you want Auto channel allocation, you don’t need to put anything in the Parallel Channel box; however, if you want your event always to execute on the same channel, use the arrow keys either side of the Parallel Channel box to dial up the figures 1-8, representing the 8 parallel channels.

When event number and channel have been set to your satisfaction, just click on ASSIGN to confirm your decision. The numbers will disappear, and your key is assigned. If you press that key while the Panel is down, then that event-number KML prog will be run.

To de-allocate an existing KML event, just click with the **right** Mouse Button over the key. Its highlight will disappear, and the key becomes unallocated.

To alter the event or channel information on an already-assigned key, just click the LEFT Mouse button over the key. The key’s Event Number and Channel Number will appear in their boxes for you to edit. Press ASSIGN when you’ve done changing the values.

See the next section, About KML, to see how you should choose which parallel channel to assign your events to.

The shaded keys, SHIFT, CTRL etc., are not available to the Key Assign Screen and may not have programs attached. ESC is reserved for calling up the Control Panel.

To leave Key Assign, press QUIT.

About KML

In this section I’ll give a complete description of the way KML works, followed by a detailed command list and notes on each function.

KML is a very simple programming language. To make it easy to learn – and, indeed, easier for Yak to write – all the commands consist of the same basic syntactic form. All KML commands consist of a single keyword followed by a number of parameters, separated by commas. For example

```
STROBE 0, 7, 7, 7
DISPLAY_FRAME 1
TRIANGLE A, B, C, D, E, F, 7
```

All are valid KML commands. Many commands don’t need any parameters at all, for example

```
RUN SEQUENCER
CLS
ISTAR
```

Are no-parameter commands.

Entering KML lines is made easy by the dint of the fact that only the **first three letters** of any KML keysword need be entered. Thus, you can enter, via the Editor, the line

```
PRO X, Y, Z, A, B
```

And, on pressing RETURN, the line will be expanded automatically to read

```
PROJECT X, Y, Z, A, B
```

In KML, you can only have one command per line; multiple statement lines are disallowed. This is in some ways a good thing; it’s important to be able to see the flow of the KML code, and multi-statement lines are often pretty opaque when you’re trying to debug some ‘code.

About the Parameters

KML uses 14-bit signed integers for numbers. This means that you can have values ranging between +/- 8192 as parameters. These numbers can be represented in a variety of ways:

CONSTANTS are just straight numbers placed directly in the parameter slots, as in, for example

```
PLOT 160,100,7
```

There is another type of constant, the **binary byte constant.** Using binary byte constants is useful in any function where a bit-pattern is required. Binary byte constants are entered by preceding an 8-bit binary value with the] symbol –

```
SYM 1,] 10100101
```

Is an example of such a constant.

VARIABLES are more flexible. You can assign a number to a variable, and once assigned, KML can alter the value of the variable and use it in subsequent commands.

Consider this code fragment:

SET 7, A	- Assigns the value 7 to variable A
LOOP	- Marks the start point of a loop (see later!)
STROBE 0,A,A,A	- Sets RGB value of colour 0 to contents of A
WAIT 3	- Waits three frametimes
SUBTRACT 1,A	- Decreases A by 1
UEQUAL –1,A	- Loops until A=-1

This bit of code, to do a screen flash, shows the use of the variable ‘A’ as a loop-counter (more on loops later!).

TYPES OF VARIABLE: There are three types of variable available to KML programs:

ALPHA VARIABLES are the most commonly used variables. These variables are represented by single letters. A-Z. Each of the eight KML parallel channels has its own set of alpha variables.

GLOBAL VARIABLES are common to **all** the parallel channels, and are so useful for passing information back and forth between programs running on different channels. They are all two-letter variables, the alphabet prefixed by the letter G, as in gA, GB, GC…GZ.

SYSTEM VARIABLES are named variables reflecting some aspect of the Trip-A-Tron system, and are used to do stuff like read the mouse and button states, or set up a starfield and suchlike. Sys vars have names of varying length; like KML commands, you only need to enter the first three letters of a long variable name and the system will expand it when you enter a line. Examples of system variables:

CX (Cursor X-position)
BUTTONS (current button-state)
ABS-ANGLE (current angle of the starfield in Yakly degrees)

Program Control Structures

There are no really heavy control structure types to learn in KML. Control is limited to simple looping and conditional branching. The basic types of command are:

LOOP
…
… - Code to execute here
…
UEQUAL A, B

The UEQUAL command in the example is just one of four loop terminators. Each will be explained in the command list, but, briefly, here they are:

UEQUAL	Until Equal
UGREATER-THAN	Until Greater Than
ULESS-THAN	Until Less Than
FOREVER	Unconditional

Loops can be nested, up to sixteen levels per KML channel. After loops, there are conditionals:

IFEQUAL A, B
…
… - Code to be executed if condition satisfied
…
ELSE ELSE is optional
…
… - Code to be executed if condition fails
…
ENDIF Conditional block delimiter

Again, there are a variety of conditional tests:

IFEQUAL
IFNOT EQUAL
IFLESS THAN
IFGREATER THAN
IFPOSITIVE
IFMINUS All pretty self-explanatory really.

Subroutines

It is possible, in KML, to call subroutines up to a depth of 18 nested subroutines. Subroutines are called by the

ROUTINE X

Command, where X is the number of the KML program you wish to call. The subroutine program must be terminated with a RETURN statement.

You can also transfer execution from one KML program to another, by using the

JUMP_TO X

Command, again with X being the number of the KML slot you wish to JUMP to.

Timing, Parallelism and All That Jazz

Now we get down to the fundamental ideas behind KML, and that which makes it different from the majority of programming languages.

First, allow me to explain the parallelism in the system. You can run up to eight KML programs ‘in parallel’ by using the KML command

PARALLEL channel-number, program-number

Of course the system isn’t *really* running the programs at the same time. What happens is this:

1: KML command language is called on 50Hz interrupt
2: KML executes statements of Program 1 until it hits a WAIT
3: KML ceases Program 1, goes to Program 2
4: KML executes 2 until a WAIT
5: -- and so on, for however many programs are running
6: KML returns from its interrupt.

Because KML sits on an interrupt, provided the processes running aren’t too processor-intensive, you can execute several KML progs ‘at once’ and still have normal Trip-A-Tron features running.

Allocation of KML Parallel Channels

There are eight parallel channels available to KML programs. On the Key Assign Screen, you can choose whether a KML program, upon being activated by the keyboard, will always use the same channel or look for a new channel. If you leave the Parallel Channel **blank (zero)**, then, when you press the key to RUN the program, KML looks through all its 8 channels and, if there’s one free, places the KML program on it and runs it. **If all channels are full, the program will not be run.**

If you dial up a specific channel, 1-8, then upon execution the program is placed on the specified channel, terminating any process which may already be using that channel. The KML PARALLEL command also ‘seizes’ the specified parallel channel from any program already using it.

You see that the ‘parallelism’ of KML comes from slices of CPU time being allocated to each of the programs running. The difference between KML programs and other multi-tasking systems is that the time allocated to each program is determined by you, the user. It all revolves around a single KML command, the WAIT statement.

Time, the Universe and the WAIT statement

KML is designed to deal with graphics effects, and for much graphical programming on the ST, the fundamental time quantum is 1/50 second, or **one frametime**. The ST’s display is re-drawn once every frametime, or 1/50 sec; provided our KML programs execute in less than one frametime, we’ll be able to use TRIP-A-TRON at the same time as KML programs are executing.

An executing KML program takes up all the CPU time when it is executing its statements. If you wrote a piece of KML code with no wait statements, then when you activated it, it would suspend all other Trip-A-Tron functions, execute, and then, once finished return control to the system.

You control the allocation of CPU time between KML programs and the rest of the system using WAIT statements. The WAIT statement takes the form

WAIT n

Where n is the number of frametimes during which the system is ti be released, to execute other KML programs and run the rest of Trip-A-Tron. It is important to use WAIT statements, because the KML interpreter runs on an interrupt, and therefore has priority over all other tasks. If you had some KML code in the following format:

```
LOOP
…
… KML statements
…
FOREVER
```

Without any WAIT statements inside the loop, the system would 'lock-up'; it would be continually executing the loop and **never** releasing the CPU to run any other tasks. What you **should** write is code like this:

```
LOOP
…
… KML statements
…
WAIT 1
FOREVER
```

This code would mean that the contents of the loop would be executed once per frametime, taking up only enough CPU time to execute the statements, then releasing the CPU for the rest of the time to carry on with normal Trip-A-Tron functions. Of course, you needn't use WAIT 1; you could use WAIT 2, WAIT 5, or whatever time value you liked.

When you're learning KML, be careful of loops without any WAIT statements. If you get into an infinite loop with no WAIT statements inside, you'll lose the system. If in doubt, bung in at least a WAIT 1 until you have the code fully debugged.

Thus, by use of the WAIT statement, you control how much CPU time each KML parallel channel gets. If you allocate more CPU time than the 1/50 sec basic time unit which runs the KML interrupt you'll get into **interrupt overrun**. The KML code will still run, but 'foreground' tasks like Trip-A-Tron pattern generation will be unable to run, because the system takes up all available CPU time with in its interrupt routine.

The WAIT statement is essential in producing animation. For example, the following code produces a cyclic 50Hz animation of the first 16 frames in the VSEQ:

LOOP
SET 0, F
LOOP
DISPLAY_FRAME F
WAIT 1
ADD 1, F
UEQUAL 16, F
FOREVER

The WAIT statement in this example controls the speed of animation. If you substituted WAIT 2 for WAIT 1, you'd get a 25Hz animation (half speed).

Now you know the basics of how KML works. Study now the command list which follows this section and the examples therein. Practise writing simple KML progs and assign them to keys and watch them run. Look at the KML programs which run the demos to see KML in action. And remember: **beware the infinite loop with no WAIT statements!**

KML Command Summary

There now follows a listing of all KML commands and system variables. Have fun.

ADD: Add a value to a variable
Syntax: ADD <VALUE>, <VARIABLE>

The ADD command is used to add a value to a specified variable. The value can take any form; the variable must be system, alpha or global.

Examples

ADD 1, X	; Add 1 to alpha var X
ADD X, GX	; Add value X to global var GX

ALINE: Line-A linedraw between two points
Syntax: ALINE <x1>, <y1>, <x2>, <y2>, <colour>

The ALINE command uses the ST Line-A call to draw a line between the points <x1>,<y1> and <x2>,<y2> in the ST colour <colour>.
The co-ordinates are clipped if they lie outside screenspace; the colour index should be in the range 0-15. The ALINE command does not use any of Main Plot, is not affected by symmetry, and does not protect foreground or background graphics.

Example:

(pretty, this)

```
CLS
SET 0, A
SET 60, B
SET 0, R
SET 1, C
LOOP
SET 2, K
LOOP
SINE A, R, X
COSINE A, R, Y
SINE B, R, I
COSINE B, R, J
ADD 160, X
ADD 160, I
ADD 100, J
ADD 100, Y
ALINE X, Y, I, J, C
ADD 3, A
ADD 3, B
ADD 1, C
SUBTRACT 1, K
UEQUAL 0, K
ADD 1, R
UEQUAL 200, R
```

AND: Logical bitwise AND operation
Syntax: AND <source value>, <variable>

This command performs an AND of the source value with the contents of the destination variable. The variable contains the result of the AND operation after execution.

Examples:

```
SET ]00001111, A
AND ]10101010,A
```

Would leave]00001010 in A.

```
SET BUTTONS, A
AND 2, A
```

Would leave A containing 2 if the left mouse button was pressed, zero if not.

Associated commands: OR, XOR

ATYPE: Specify pattern action type
Syntax: ATYPE <pat-mode>

The ATYPE command allows you to change the pattern generation mode between decay, line and expandor modes. The pat-mode parameter should be 0 for decay, 1 for line, or 2 for expandor types. Note that the change will not take effect until the next WAIT statement.

Examples:

```
ATYPE 2
WAIT 1
```

Set Expandor mode

```
LOOP
ATYPE 0
WAIT 6
ATYPE 1
WAIT 2
FOREVER
```

Generates a hybrid decay/line mode

Associated system variable: PATTERN

BLOCK: Draw a solid block of colour on the current screen
Syntax: BLOCK <x1>, <y1>, <x2>, <y2>, <colour>

BLOCK draws a solid rectangle of the specified colour on the current screen. The top-left and bottom-right corners are determined by the co-ordinates <x1>,<y1> and <x2>,<y2>. The colour index is a standard ST colour index, 0-15. The BLOCK command uses line-A and as such does not use the Main Plot routine, respond to symmetry, or respect background or foreground graphics.

Example of BLOCK:

```
CLS
SET 0, A
SET 100, B
SET 319, X
SET 101, Y
SET 1, C
LOOP
BLOCK A, B, X, Y, C
ADD 1, A
ADD 1, Y
SUBTRACT 1, X
SUBTRACT 1, B
ADD 1, C
UEQUAL 0, B
```

Associated commands: BOX

BOX: Draw an outline box on the current screen
Syntax: BOX <x1>, <y1>, <x2>, <y2>, <colour>

The BOX command draws an outline rectangle of colour ‘colour’ on the current screen with <x1> and <y1> giving one corner, <x2> and <y2> giving the diagonally-opposing corner. BOX uses the a-line, and so stomps on any foreground/background graphics and is not subject to symmetry or the Main Plot Routine.

Example:

A simple box-diddler program.

```
CLS
CURSOR 1, 0, 1, 0
SET 0, A
SET 0, B
LOOP
SET BUTTONS, N
ADD 2, N
IFNOT-EQUAL 0, N
SET 7, C
SET CX, X
SET CY, Y
LOOP
BOX A, B, X, Y, C
SUBTRACT 1, C
SUBTRACT 1, X
SUBTRACT 1, Y
ADD 1, A
ADD 1, B
UEQUAL 0, C
SET X, A
SET Y, B
ENDIF
WAIT 1
FOREVER
```

Associated commands: BLOCK

CGET: Get palette colour RGB components
Syntax: CGET <palette-#>, <colour-#>, <r-var>, <g-var>, <b-var>

CGET returns the RGB value of a given colour in a given palette. The palette number 1-200 is passed in the <palette-#> parameter, the colour number in the <colour-#> parameter; the R, G and B values are returned in <r-var>,<g-var> and <b-var> respectively.

Example:

```
CGET 20, 5, R, G, B
```

Will leave R, G and B holding the red, green and blue component values making up colour #5 in palette #20

CHANNEL: Activate/de-activate colour channels
Syntax: CHANNEL <channel-#>, <status>

The CHANNEL command allows you to turn off and on the colour channels. These channels, numbered 1-9, hold single, static palette definitions; if more than one channel is activated, then the palette definitions will be multiplexed, allowing you to do simple colour mixing in order to expand the ST's colour range. The <channel-#> parameter specifies which channel to turn on or off; the <status> parameter should be set to 1 to turn ON a channel, and 0 to turn OFF a channel. ANY open colour channel will over-ride any PALETTE settings.

Examples:

```
CHANNEL 1, 0
```

Turn off channel 1

```
SET 1, C
LOOP
CHANNEL C, 1
ADD 1, C
UEQ 10, C
```

Turns on all nine channels.

CLS: Clear the current logical screen
Syntax: CLS

CLS is a straightforward Clear Screen command. It fills the whole screen with colour zero. It applies to the current Trip-A-Tron screen.

Example:

```
SET 0, F
LOOP
LOGICAL_SCREEN F
CLS      ; Clear the screen
ADD 1, F
UEQUAL 10, F
LOGICAL_SCREEN –1
```

Clears the first ten frames of VSEQ.

Associated commands: LOGICAL SCREEN

COLOUR: Set SillyScope trace colour wave or constant
Syntax: COLOUR <trace>, <-1>, <const>, <0>, <0> **for constant**
COLOUR <trace>, <wave>, <start>, <speed>, <amplitude> **for an oscillator.**
(see the section on XSWEEP for an explanation of SillyScope syntax and full example)

COSINE: Return scaled cosine value
Syntax: COSINE <angle>, <scalar>, <variable>

The COSINE command isn't like (say) COS in Basic, which returns a floating-point value. Because we are working with integers, COSINE behaves differently. The <scalar> value specifies the peak-to-peak amplitude of the wave; the wave value at <angle> is calculated and returned as the contents of <variable>. (The angle is specified by values of 0-255, rather than 0-360 as in 'conventional' maths). That sounds weird, but actually it's easy to understand. Check out the SINE example to see what I mean.

Associated commands: SINE

CUBE: Set enabled faces for Cubic Remap
Syntax: CUBE <face 1>, <face 2>, <face 3>

The CUBE command allows you to select which of the possible three cube faces to draw on when in Cubic Remap mode. The three parameters should be 1 to enable a face, 0 to disable a face, and relate to the cube surfaces as follows:

Face 1	governs the top face of the cube
Face 2	the left-hand face
Face 3	the right-hand face

This command is 'dirty'; if executed while pattern-generation is occurring, stray pixels may be left behind. Follow it with a CLS command for safety.

Examples

```
CUBE 1, 1, 1
```

Turns on all three cube faces

```
CUBE 1, 1, 0
```

Turns on the top and left-hand face

CURSOR: Set cursor attributes
Syntax: CURSOR <curs-#>, <on-off>, <viz>, <trail>

This command allows you to turn on or off either of the two cursors, and specify whether they are visible and (in the case of the secondary cursor) whether to trailback or not. The <curs-#> parameter should be 1 (for main cursor) or 2 (for secondary cursor); <on-off> is zero for Off, 1 for On; <viz> is 0 for invisible, 1 for Visible; and <trail> is 0 for non-trailback and 1 for trailback mode (secondary cursor only; this parameter can be anything for primary cursor, and has no effect).

Examples:

```
CURSOR 2, 1, 0, 1
```

Turn on cursor 2, leave it invisible, and in Trailback Mode

```
CURSOR 1, 1, 0, 0
```

Make the main cursor invisible.

DISPLAY_FRAME: Show a VSEQ frame
Syntax: DISPLAY_FRAME <frame-number>

The DISPLAY_FRAME command switches the physical screen base so that it points at the specified frame in the video sequencer. Its main use is in animation. The logical screen base remains pointed at the main Trip-A-Tron display screen. The VSEQ frame will be displayed until another DISPLAY_FRAME command is issued, or until the Control Panel is raised up. To restore the normal Trip-A-Tron screen, issue the command DISPLAY_FRAME –1. The DISPLAY_FRAME command alters the Current Palette to that of the frame chosen, unless a LOCK command has been issued.

Example:

To display an animation consisting of the first 16 frames of VSEQ at 25Hz, then restore the Trip-A-Tron display:

```
SET 0, F
LOOP
DISPLAY_FRAME F
WAIT 2
ADD 1, F
UEQ 16, F
DISPLAY_FRAME –1
```

Associated commands: PICTURE, UNPACK, LOCK, UNLOCK

DIVIDE: Divide a variable by a number
Syntax: DIVIDE <value>, <variable>

DIVIDE performs an integer division of the contents of the variable by the value. The value can take the form of a numeric literal, or a variable.

Example:

```
SET 50, A
DIVIDE 6, A
```

Would leave A containing 8 (the remainder is discarded)

Associated commands: MULTIPLY, MOD

ELLIPSE: Draw an ellipse or circle on the current screen
Syntax: ELLIPSE <x-co>, <y-co>, <x-rad>, <y-rad>, <colour>

Draws an ellipse or circle on the screen. The <x-co> and <y-co> parameters specify the centre of the generated ellipse; the <x-rad> and <y-rad> parameters specify the x-radius and y-radius of the ellipse respectively: finally, <colour> gives the colour (surprise surprise). The function ELLIPSE is a GEM function; it doesn't use my zarjaz Plot Routine, so none of the settings or symmetries apply; and it zaps PICTURE graphics if any are present.

Example:

Enter this and activate it, then fiddle with y'mouse:

```
CLS
SET 0, A
SET 1, C
```

```
LOOP
SINE A, 50, R
SET 320, X
SUBTRACT CX, X
ELLIPSE CX, CY, R, R, C
ELLIPSE X, CY, R, R, C
ADD 5, A
ADD 1, C
IFEQUAL 16, C
SET 1, C
ENDIF
WAIT 1
FOREVER
```

ELSE: Optional conditional block marker
Syntax: ELSE

The ELSE statement is used to add extra flexibility to the conditional block structure. Normally, the code inside a conditional block is executed only if the condition is true; if the condition is false, execution jumps past the block, to the ENDIF statement, as follows:

```
IF <condition>
…
… - The code executed if condition is true
…
ENDIF
… - This code is executed always
```

As you can see, if the condition fails it skips to the ENDIF and the following code is executed. However, you may want to supply a block of code to be executed **only** if the condition fails. The ELSE statement allows you to do this. If a condition fails, the KML interpreter first looks for an ELSE statement. If one is found, the code following it is executed. If no ELSE is found, execution is transferred to the code following the ENDIF. For example:

IF <condition>
…
… This code executed if condition is true
… And on reaching ELSE, jumps to ENDIF
ELSE
…
… - This code **only** executed if condition is FALSE
…
ENDIF
… This code always executed

Associated commands: any IF command, ENDIF.

ENDIF: Conditional block terminator
Syntax: ENDIF

The ENDIF statement is used to terminate a conditional block. It must be used, and is not optional. If the condition specified in the originating conditional statement is **not** satisfied, then program execution passes to the statement following ELSE (if any ELSE statement is present); if no ELSE is present then execution passes to the command following the ENDIF statement.

Example of ENDIF with no ELSE:

```
If <condition>
…
… - Code here executed if condition is true
…
ENDIF
…  - Execution skips to here if condition is false
```

Associated commands: any IF command, ELSE

EXTERN: Execute a KML program on a remote machine
Syntax: EXTERN <node #>, <prog #>, <par chan #>

The EXTERN command is used when you have two or more Ataris running Trip-A-Tron linked via MIDI. The first parameter is the number of the Atari you want to send the command tol the second parameter is the program number to run on the remote system's KML interpreter; the third parameter tells the remote system which parallel channel to use (a zero parameter here means 'look for a free channel and use that if available'). Legal node numbers are in the range 1-8.

Example:

```
EXTERN 2, 24, 6
```

Will make the remote system 2 run its KML program #24 on parallel channel 6.

Associated commands: SEND, UNSEND, LINK

FOREVER: Unconditional loop closure
Syntax: FOREVER

The FOREVER loop closure causes an unconditional branch back to the beginning of the LOOP. You should take care you have at least one WAIT statement inside a forever-LOOP, or else you could lock up the system. Provided that you have a WAIT statement somewhere inside the loop, FOREVER loops can be terminated by raising the Control Panel, or by another program taking over the loop's parallel channel.

Example:

```
LOOP
ROTATE-PALETTE 1, 15, -1
WAIT 2
FOREVER        ; Continuously rotate the current palette
```

Associated commands: LOOP

FXENABLE: Enable the in-line plot routine effects
Syntax: FXENABLE <binary byte>

The FXENABLE command is used to turn on or off the inline effects of the main PLOT routine. The binary byte pattern parameter specifies whilch effects are turned OFF and ON by the command.

Bit 1 = Pre-sym
Bit 2 = Remap
Bit 3 = Cubic map
Bit 4 = Shear
Bit 5 = Resize
Bit 6 = Pos-sym

Examples:

```
FXENABLE ]00100101
```

Enables pre-sym, cubic remap and post-sym

```
FXENABLE ]00100010
```

Enable remap and post-sym

If no .MAP file is in memory, remap will not be activated.

Associated commands: FXSTATE

FXSTATE: Return current FX enabled information
Syntax: FXSTATE <variable>

This command returns an eight-bit binary byte pattern representing the currently-enabled functions of the Main Plot Routine. The bits in the binary pattern are laid out as for the FXENABLE command.

Example:

```
FXSTATE A
AND ]00101111, A
FXENABLE A
```

Would disable just the Resize option, leaving undisturbed any other Main Plot Routine functions.

Associated commands: FXENABLE

GATE: Trigger a one-shot waveform for oscillator FX
Syntax: GATE <wave-#>

The GATE command is used only in conjunction with the Oscillator FX (see the manual section on Oscillator FX for more details). If a waveform has been set into One-Shot mode, it remains inactive until it is triggered by a KML GATE command. The parameter specifies which waveform to trigger, and lies in the range 1-8.

Example:

```
GATE 5
```

Would trigger a one-shot of Waveform #5., All other effects using waveforms such as SillyScope are unaffected by this command.

GFORCE: Get optional Force parameter on current KML track
Syntax: GFORCE <variable>

GFORCE is used in conjunction with the internal sequencer. Events entered into the sequencer may have an optional Force parameter specified with each event. The Force parameter is similar to a MIDI Velocity parameter; it is entirely optional and can be ignored if you don't want to bother with it. If you do want to use it, then the GFORCE command is used to get the current FORCE value of the KML channel which is executing the GFORCE command. The Force value is returned as the contents of the specified variable, and lies in the range 0-31.

Example:

```
GFORCE A
```

On KML channel 1 would return the channel's Force setting in A.

GLOBAL: Change to a new Global Parameter Bank
Syntax: GLOBAL <bank #>

The GLOBAL command allows you to switch between the sixteen possible global parameter banks. These settings are made from the Global Parameters Setup Screen on the main control panel. Changing the GLOBAL parameters in mid-pattern-generation may, under certain circumstances, leave 'dead pixels' on the screen, so you may want to follow the command with a CLS.

Example:

```
GLOBAL 3
CLS
```

Switches in global parameter bank #3, and is tidy about it.

IFEQUAL: Conditional block delimiter
Syntax: IFEQUAL <value 1>, <value 2>

If value1 and value2 are equal, then control is passed to the statement following the IFEQUAL command. If the values are not equal, the program skips to the statement after the ELSE command (if present) or after the ENDIF command (if no ELSE is present).

Example:

```
IFEQUAL A, B
…
… - Code to be executed if A=B
…
ELSE
…
… - Code to be executed if A <> B
…
ENDIF
```

Associated commands: ELSE, ENDIF

IFGREATER_THAN: Conditional block delimiter
Syntax: IFGREATER_THAN <value 1>, <value 2>

If <value 2> is greater than <value 1>, execution passes to the statement following the IFGREATER than command. If not, execution passes to the next ELSE statement, or to ENDIF if no ELSE is present.

See IFEQUAL for example.

Associated commands: ELSE, ENDIF

IFLESS_THAN: Conditional block delimiter
Syntax: IFLESS_THAN <value 1>, <value 2>

If <value 2> is less than <value 1>, execution passes to the statement following the IFLESS_THAN command. If not, execution skips to the following ELSE statement (if present) or ENDIF (if no ELSE).

For example, look at IFEQUAL example.

Associated keywords: ELSE, ENDIF

IFMINUS: Conditional block delimiter
Syntax: IFMINUS <value>

If <value> is negative, the code immediately following the IFMINUS statement is executed. If <value> is positive, control passes to the next ELSE statement (if present) or to the ENDIF statement (if no ELSE).

For a general example of conditionals, see IFEQUAL

Associated keywords: ELSE, ENDIF

IFNOT_EQUAL: Conditional block delimiter
Syntax: IFNOT_EQUAL <value 1>, <value 2>

The code after the IFNOT_EQUAL statement is executed only if <value 1> and <value 2> are NOT equal. If they are equal, execution skips to the next ELSE statement (if present) or ENDIF statement (if no ELSE is given).

See IFEQUAL for example

Associated commands: ELSE, ENDIF

IFPOSITIVE: Conditional block delimiter
Syntax: IFPOSITIVE <value>

If <value> is positive, the code after the IFPOSITIVE statement is executed, if <value> is negative, execution jumps to the next ELSE statement (or ENDIF if no ELSE is given).

For general conditional-block example see IFEQUAL

Associated commands: ELSE, ENDIF

ISTAR: Force a star to be created
Syntax: ISTAR

This is used for advanced starfield making. Normally, creation of individual stars is governed by the INRATE system variables. You can turn off automatic star generation by setting INRATE to –1; you can then generate stars individually, whenever you want, by issuing an ISTAR command. You might want to do this in order to make more than one star appear in a single frametime (which isn't possible under automatic starfield generation).

Example:

```
SET –1, INRATE
SET 0, X
LOOP
SET X, INX
ISTAR
ADD 10, X
UEQUAL 320, X
```

This code fragment would create a line of 32 stars, all during one frametime.

JUMP_TO: Jump to another KML routine
Syntax: JUMP_TO <KML program #>

This command just transfers the flow of program execution to the program specified in the parameter. If the parameter is not a numeric literal, ensure that it lies in the range 1-128.

Example:

```
JUMP_TO 30    ; Jump to beginning of program #30
```

KILLALL: Cease execution on all KML channels
Syntax: KILLALL

The KILLALL command will, **at the next WAIT statement**, stop the execution of all the KML programs currently on the parallel channels. Statements after the KILLALL statement are executed, but only up to the next WAIT statement, and only on the channel issusing the KILLALL command.

LASER: Set SillyScope channel laser mode on/off
Syntax: LASER <channel-#>, <mode>

This command is used to set the mode type for SillyScope traces. The first parameter <channel-#> should be 1 or 2 depending on which trace you're setting up; the second parameter should be 0 (for dot-mode) or 1 (for laser-mode).

Note that the LASER command is 'dirty'; if you switch modes without first switching off the channel and WAITing 1, pixels or vectors will be left on the screen.

The correct way to switch modes would be:

```
SCOPE 0, -1
WAIT 1
LASER 1, 1
SCOPE 1, -1
```

To change scope trace #1 from dot to laser.

Associated commands: SCOPE, XWAVE, YWAVE, XPHASE, YPHASE, XAMPLITUDE, YAMPLITUDE, XCENTRE, YCENTRE, SCTRAIL, COLOUR

LINK: Slave to a remote system's mouse commands
Syntax: LINK <node #>

The LINK command is used by a remote system to slave it to the mouse movements of a remote master system. The master system should first be placed in SEND mode, then the remote system should execute LINK, with the parameter being the node number of the master system (usually 1). To unlink a slaved system, it should execute LINK with a parameter of –1 to sever the link.

Examples:

LINK 1

Slave to mouse movements of remote system 1

LINK –1

Restore independent operation

Associated commands: SEND, UNSEND

LLAMA: Set Line Length parameter for line draw
Syntax: LLAMA <line length>

The LLAMA command sets the Maximum Line Length parameter to Line Mode lines. I could have called the command LINELENGTH I suppose, but I like llamas, so what the heck. The single parameter following the command gives the new line length.

Example

LLAMA 50

Sets the maximum possible line-mode line length to 50 pixels.

LOCK: Prevent subsequent palette changes
Syntax: LOCK

LOCK protects the current palette number from being changed by operations which would normally alter it. For example, normally, while flicking through VSEQ screens to generate animation, each DISPLAY_FRAME command would cause each frame's palette to become the current palette. Using LOCK you can prevent this from happening, and run the animation with the palette of your choice.

Example:

Run a 30-frame anumation under palette #100

```
PALETTE 100, -1
LOCK
SET 0, F
LOOP
DISPLAY_FRAME F
WAIT 2
ADD 1, F
UEQUAL 30, F
```

Associated commands: UNLOCK, PALETTE, DISPLAY_FRAME

LOGICAL_SCREEN: Change the Trip-A-Tron logical screen base
Syntax: LOGICAL_SCREEN <VSEQ frame #>

The LOGICAL_SCREEN command tells Trip-A-Tron to use a different area of memory for its screen. You can set up the system so that Trip-A-Tron pattern-generation, starfields, SillyScope stuff and KML CLS and various drawing commands occur directly onto the VSEQ frame specified in the LOGICAL_SCREEN command. If the frame pointed to in the command has its own palette already designated, then that palette will become the current palette. If no palette exists in that frame slot, the current palette is copied into it. By using LOGICAL_SCREEN on successive VSEQ frames, and using KML drawing commands, it becomes possible to build simple animations. Specifying –1 as the frame parameter restores the normal Trip-A-Tron screen.

Example:

LOGICAL_SCREEN 2

Set Trip-A-Tron commands to happen on KML frame 2

LOGICAL_SCREEN –1

Set Trip-A-Tron normal screen back on

Associated commands: CLS

LOOP: Delimit the start of a program loop
Syntax: LOOP

The LOOP command is used to denote the starting point of a program loop. The LOOP command needs a loop-terminator to determine how much code is to be LOOPed around, and how many times to execute it. LOOPs can be nested up to 16 levels.

Examples:

```
LOOP
WAIT 1
MVELOCITY 1, A
UEQUAL 0, A    ; Waits for a key to be pressed on an external
               ; synthesiser on the MIDI link
SET 0, X
LOOP
SET 0, Y
LOOP
PLOT X, Y, 8
ADD 10, Y
UEQUAL 200, Y
ADD 10, X
UEQUAL 320, X          ; Draws a grid of dots on the screen
```

Associated commands:

UEQUAL, UNEQUAL, UGREATER_THAN, ULESS_THAN, FOREVER

MACRO_LOAD: Load all the bits in the current macroload file
Syntax: MACROLOAD

The MACRO_LOAD command tells the system to look for and LOAD all the files listed in the current Macro file. The files are loaded in and execution of KML stops (it has to, since the KML program issuing the MACRO_LOAD command may itself be loaded over!) Since the very last thing that a MACRO_LOAD command looks for is another MACRO file, you can set up a situation where programs can chain each other (just look at the two Demo files to see that in action). See the section on making macro files for more details.

MAP: Select between loaded maps
Syntax: MAP <map-no>

The MAP command is one for the memory-rich amongst you; it only really comes into any use if you have more than one .MAP file loaded. The command simply allows the switching between different MAP files. If you haven't the .MAPs loaded, the command is ignored.

Examples:

MAP 1

Selects Map Number 1

MAP 3

Selects Map Number 3

This command is 'dirty'; if you change maps while PATTERN GENERATION is going on, stray pixels may be left. Follow the command with a CLS to be clean.

Associated commands: MCLEAR, MSET

MBEND: Pick up MIDI Pitch Bend information
Syntax: MBEND <channel-#>, <variable>

Using the MBEND command, you can check for information sent by the operation of an external synth's Pitch Bend wheel. You specify the MIDI channel on which to look for the information, and a variable to receive the pitchbend value, which will be in the range 0-127.

Example:

MBEND 1, A

Will return the current pitchbend setting in the variable A

MCHAN: Set 'MyChannel' setting for a networked Trip-A-Tron
Syntax: MCHAN <node-#>

The MCHAN command allows a remote Atari to set its own node number when using multiple networked Trip-A-Trons linked via MIDI. To set a machine's node number, just issue MCHAN followed by the machine's node number.

Example: MCHAN 2

Sets the machine to be node #2

Associated commands: SYNTH, EXTERN

MCLEAR: Clear the currently-selected map
Syntax: MCLEAR

MCLEAR will zero a currently-loaded .MAP file, or if no file is loaded, MCLEAR will allocate itself room in VSEQ and set it up as a MAP. If insufficient VSEQ memory is available, the command is ignored.

MCLEAR is usually used in conjunction with subsequent MSET commands to generate a new Remap file.

If one or more MAP files are in memory, then MCLEAR will zero whichever one is the currently-selected MAP.

Associated commands: MAP, MSET

MOD: Return the remainder of a division operator
Syntax: MOD <value>, <variable>

The MOD command returns the remainder part of an integer division. This is useful for particularly in truncating unruly variables to fit into certain ranges.

Example:

```
SET 34, A
MOD 16, A
```

Would leave A containing 2, since 34 divided by 16 is 2 remainder 2.

Associated commands: DIVIDE, MULTIPLY

MORPH: Generate intermediate frames between two maps
Syntax: MORPH <start map>, <end map>, <s-frame>, <d-frame>, <#-frames>

The MORPH command is the nub-end of an experimental VSEQ function which I didn't leave in the VSEQ section in the end, as it requires forbidding amounts of memory to run, and is only really of any use to those of us fortunate souls with Mega ST4s. What it does is: take two MAPs and a picture; start with the pic wrapped around one MAP, and then calculate the 'in-between' phases of the picture as it flows and ends up wrapped around the second MAP. (You can, in fact, use MORPH with only one map; by specifying a zero as a MAP number, the command will use the standard flat screen in place of a MAP). I expect I'll fully-integrate this command into future expanded-memory versions of Trip-A-Tron, but rather than discard it utterly I've left this command in, so the memory-rich amongst you can have a go.

Examples:

(Assuming two MAPs loaded)

```
MORPH 0, 1, 0, 1, 10
```

Generates 10 frames showing the transition of the image in VSEQ frame 0 onto MAP number 1. Places the frames starting at VSEQ frame #1.

```
MORPH 1, 2, 20, 40, 30
```

Generates thirty frames, starting at frame 40, of the image in frame 20 morphing from map 1 to map 2.

MPITCH: Get MIDI pitch data on a specified channel
Syntax: MPITCH <channel #>, <variable>

This command gets the MIDI pitch data from the specified channel. (For a detailed explanation of how the CHANNEL parameter changes according to SYNTH or POLY mode, see the description under the MVELOCITY command heading).

Example:

Program to wait for a synth keypress on MIDI channel 1 and return its pitch value in alpha variable P.

```
SYNTH 1
LOOP
MVELOCITY 1, A
WAIT 1
UNEQUAL 0, A
MPITCH 1, P
```

Associated commands: MVELOCITY, SYNTH, POLY, MBEND

MSET: Generate a mapping point in a REMAP file
Syntax: MSET <orig-x>, <orig-y>, <remap-x>, <remap-y>

This command generates the remap information for one screen co-ordinate pair. You supply the original point in the co-ordinates <orig-x> and <orig-y>, and the transformed co-ordinates in <remap-x> and <remap-y>. This command only works if there is .MAP space allocated in VSEQ ram, either by the loading of a .MAP file or by previous use of MCLEAR statement.

Examples:

```
MSET 100, 100, 50, 36
```

Would cause the point (100,100) to be mapped onto the point (50, 36) in a subsequent REMAP plot.
Now, if you have at least eight frames of VSEQ free, enter and run the following prog-ette:

```
CLS
FXENABLE 0
MCLEAR
SET 0, Y
LOOP
SET 0, X
SET 0, Z
SET 0, A
LOOP
SINE A, 50, I
ADD X, I
SET Y, J
SUBTRACT 100, J
PROJECT I, J, Z, Q, R
ADD 100, R
PLOT Q, R, Y
MSET X, Y, Q, R
ADD 1, A
ADD 1, X
ADD 4, Z
UEQUAL 320, X
ADD 1, Y
WAIT 1
UEQUAL  200, Y
```

Execute the function, then when it has finished, raise the panel and click the right Mouse Button over the Remap icon, to see the grid projected on the remap just created. You SAVE such generated mappings as .MAP files, they're BIG, 256K each, but they do enable fast remapping onto complex shapes!

Associated commands: MAP, MCLEAR

MULTIPLY: Multiply two numbers
Syntax: MULTIPLY <value>, <variable>

The contents of <variable> are multiplied by the <value>

Example:

```
SET 8, A
MULTIPLY 11, A
```

Would leave A holding the value 88.

Associated commands: DIVIDE, MOD

MVELOCITY: Get MIDI velocity data on specified channel
Syntax: MVELOCITY <channel #>, <variable>

Used in conjunction with an external synthesiser on MIDI. In normal SYNTH mode, the system listens to all 16 MIDI channels, but can only pick up one note on each. In POLY mode, up to 16 simultaneous key closures can be detected on any one MIDI channel. Depending on whether the system is in SYNTH or POLY mode, the <channel> variable has different meanings.

In SYNTH mode, <channel> refers to the MIDI channel number.

In POLY mode, the MIDI channel is already specified by the POLY command. The <channel> parameter now refers to the note number. In POLY mode, the first key held down is note #1, the second key is note #2, and so on, up to a maximum note #16. As notes are released, their velocity value is set to zero.

The <variable> is a variable in which the velocity value will be returned.

Example:

```
SYNTH 1
LOOP
MVELOCITY 1, A
WAIT 1
UEQUAL 0, A
```

Waits for a key to be pressed on MIDI channel 1.

Associated commands: MPITCH, MBEND, SYNTH, POLY

MWAVE: Modify waveform value
Syntax: MWAVE <wave #>, <entry #>, <new value>

MWAVE allows you to construct your own waveforms in one of the eight system waveform slots. You might use MWAVE commands to generate special waveforms which are unobtainable using the standard waveform-edit screen. The <wave #> parameter specified which wave to alter; the <entry #> parameter specified which wave table entry to modify (there are 256 entries per wave); and <new value> is the value to place in the wave table and should be a signed byte value (+/- 127).

Example:

```
SET 0, C
LOOP
RND 256, V
MWAVE 1, C, V
ADD 1, C
UEQUAL 256, C
```

Will generate a ‘noise’ waveform in system wave slot 1.

Associated commands: USER_WAVE

NEGATE: Change the sign of a value
Syntax: NEGATE <variable>

NEGATE simply changes the sign of the specified variable. That is to say:

```
SET 6, A
NEGATE A
```

Would leave A holding the value –6.

NUMERIC: Display a number at specified co-ordinates
Syntax: NUMERIC <x-co>, <y-co>, <value>

NUMERIC is most useful as a debug command, when you need to see the value of a variable. The three parameters are the x-position and y-position of where to display the number, and the value itself. The number is XORed onto the display, and so can be removed with a subsequent, identical NUMERIC command.

Example:

```
LOOP
NUMERIC 0, 190, CX
NUMERIC 25, 190, CY
SET CX, X
SET CY, Y
WAIT 10
NUMERIC 0, 190, X
NUMERIC 25, 190, Y
FOREVER
```

Displays a continuously-updating mouse-position in the lower left hand corner of the screen.

OR: Perform logical-OR operation on a value
Syntax: OR <source value>, <variable>

A logical-OR operation is performed on the contents of the variable, using the source data, which may be in any format. The resultant value is returned as the contents of the variable.

Example:

```
SET ]00001111, A
OR ]10100000, A
```

Would leave A containing the value]10101111

Associated commanbds: AND, XOR

PALETTE: Select a palette from the palette store
Syntax: PALETTE <pal-no>, <intensity>

The PALETTE command can achieve two different things. In its usual form, the <intensity> parameter is set to –1. In this form, the command means ‘make palette number <pal-no> the new current palette’. If the command is issued with the <intensity> parameter set to a value between 0-7, then the command means ‘take the colours from palette number <pal-no>, and copy them to the current palette at a level of intensity specified by <intensity>’.

Examples:

```
PALETT 40, -1
```

Make palette 40 the current palette

```
PALETTE 100, -1
PALETTE 40, 4
```

Make palette 100 the current palette, then copy palette 40 to palette 100 at half-intensity.

Associated commands: ROTATE_PALETTE, STROBE, LOCK, UNLOCK

PARALLEL: Launch a program on specified Parallel Channel
Syntax: PARALLEL <channel #>, <program #>

The PARALLEL command is used to start the specified program number (1-128) running on the specified channel (1-8). The two parameters may be specified in any form, as literal numerics or variables, but ensure that any variables lie within the correct ranges, i.e. 1-128 for KML prog number, and 1-8 for channel number. Any process already occupying the specified channel will be replaced by the given program. Execution of the new program will begin immediately.

Example:

(A nice example this. YAK's Official Who-Needs-An-Amiga-To-Do-Multitasking-Demo).

Enter the following prog into KML slot #1

```
FXENABLE 0
CLS
PARALLEL 2, 2
PARALLEL 3, 3
LOOP
RND 160, X
RND 100, Y
RND 15, GC
ADD 80, X
ADD 50, Y
ADD 7, GC
PLOT X, Y, GC
WAIT 1
FOREVER
```

Then, into KML slot 2, type:

```
LOOP
RND 160, X
RND 160, A
RND 100, Y
RND 100, B
ADD 240, X
ADD 240, A
ADD 50, Y
ADD 50, B
ALINE X, Y, A, B, GC
WAIT 1
FOREVER
```

- and into KML slot 3:

```
LOOP
RND 320, X
RND 320, A
RND 100, B
ADD 160, X
ADD 160, A
ADD 150, Y
ADD 150, B
BLOCK X, Y, A, B, GC
WAIT 1
FOREVER
```

Okay, now go to Key Assign and assign KML event 1 to the spacebar, specifying Parallel Channel 1. Quit Assign, drop the panel and push SPACE to see all three programs running at once.

Associated commands: KILLALL

PATTERN: Initialse a Trip-A-Tron pattern
Syntax: PATTERN <x-position>, <y-position>

This command attempts to start a Trip-A-Tron pattern at the specified X, Y position. It behaves exactly as if you pressed the Mouse button at that position on the screen. You can use the PATTERN command to tweak existing Trip-A-Tron mode.

Example:

```
SET 0, S
LOOP
SET BUTTONS, V
ADD 2, V
IFNOT_EQUAL 0, V
SINE S, 50, A
COSINE S, 50, B
ADD CX, A
ADD CY, B
PATTERN A, B    ; This wee prog-ette uses the PATTERN
ENDIF           ; command to add an extra ‘orbiting’
ADD 6, S        ; pattern to the main cursor
WAIT 1
FOREVER
```

PICTURE: Merge a VSEQ frame with the main Trip-A-Tron screen
Syntax: PICTURE <frame #>, <priority>, <action>

This command allows you to place a picture either behind of, or in front of, the main Trip-A-Tron pattern screen so that generated patterns fall either on top of, or behind, the image as you perform. The parameters are constructed as follows: <frame #> specifies which frame of VSEQ to take the picture from; <priority> specifies whether the image will be in front of or behind Trip-A-Tron patterns (0 = behind, 1 = in front of); <action> specifies whether the sysem will immediately draw the picture onto the Trip-A-Tron screen, or whether it will wait, ‘filling-in’ the picture as you draw pattern (try it, it’s weird). Specifying a frame number of –1 will disable the picture merge.

Examples:

```
PICTURE 0, 1, 1
```

Place frame #0 in the foreground, straight away

```
PICTURE 3, 0, 1
```

Place frame #3 in the background, straight away

```
PICTURE –1, 0, 1
```

Remove any picture currently displayed, leaving a normal, clear Trip-A-Tron screen.

Associated commands: DISPLAY_FRAME

PLOT: Draw a pixel on the current screen
Syntax: PLOT <x-co-cord>, <y-co-ord>, <colour>

PLOT is a fundamental Trip-A-Tron draw command, and has been used in a lot of the example programs in this command list. Using PLOT is very straight forward; just pass the three parameters as follows: <x-co-ord> a value between 0 and 319; <y-co-ord> a value between 0 and 199; <colour> as a regular ST colour index between 0 and 15. PLOT passes through the main plot routine for Trip-A-Tron, and as such is subject to any remaps, symmetries and suchlike you may have enabled. If you just want to do straight PLOTing with no symmetry, just put a

```
FXENABLE 0
```

At the start of your routine to ensure that all Main Plot stuff is turned off.

Examples:

```
PLOT 100, 100, 15
```

Plots a pixel on the main image screen, in colour #15.

```
FXENABLE 0
CLS
SET 0, R
SET 0, A
SET 0, C
```

```
LOOP
SINE A, R, X
COSINE A, R, Y
ADD 160, X
ADD 100, Y
PLOT X, Y, C
ADD 1, C
ADD 1, R
ADD 2, A
UEQUAL 300, R
```

Draws a spiral on your screen

POLY: Set MIDI external synth reception to POLY mode
Syntax: POLY <midi-chan #>

The POLY command allows you to look for multiple notes across the specified MIDI channel. In POLY mode, the system can detect up to sixteen simultaneous notes held down on an external MIDI keyboard. The <channel #> parameter in the MVELOCITY and MPTICH commands becomes an index to the note number. A single note played has note number 1; if two notes are held down, they can be read as note number 1 and note number 2 respectively. To effectively use POLY mode you need to scan the 16 possible note values for notes which are 'in use' (the MVELOCITY value will be non-zero).

Example:

```
SYNTH 1           ; Look for external synth
POLY 1            ; check for notes on MIDI ch. 1
LOOP
SET 1, N
LOOP
MVELOCITY N, V; look for a note
IFNOT_EQUAL 0, V
SET N, P
ADD 1, P
SET N, K
ADD 9, K
PARALLEL P, K
ENDIF
ADD 1, N
UEQUAL 7, N
WAIT 1
FOREVER
```

This program will, if run on Par. Channel 1, read an external synth keyboard and detect up to 7 notes of polyphony. If a note is detected, a KML program in the range 10-16 will be run on channel 2-8, depending on the note number of the note detected (for example KML event #16 would only be activated if all 7 notes were held down).

Associated commands: SYNTH, MCHAN

PROJECT: Project a 3-D coordinate into 2-D
Syntax: PROJECT <x-co>, <y-co>, <z-co>, <x-2D>, <y-2D>

Provide the co-ordinates of a point in three-dimensions, and this function will return the 2-D screen co-ordinates of its projection onto the screen. The projection depth, by the way, for this function may be altered by setting the starfield system variable PROJECT DEPTH. Th 2-D co-ordinates returned by this call are centred around the point (0, 0) so if you want to have the results centred on the screen, use

```
ADD 160, x
ADD 100, y
```

To centre up.

Here's a qute complex example, which generates a SineScape

```
CLS
FXENABLE ]000100000
SYM 0, ]00000101
SET 1, C
SET –400, X       ; 3D start point x=-400
SET 0, D          ; sine D used to modulate successive waves
LOOP
```

```
SET 0, Z              ; display made of sine waves receding from z=0
SET 0, A              ; angle for wave
LOOP
COSINE D, 500, H          ; H = height of current wave
SINE A, H, Y
ADD 150, Y            ; shift it down for extra perspective
PROJECT X, Y, Z, I, J   ; 2D in I, J
ADD 160, I
ADD 100, J            ; centre on (160, 100)
PLOT I, J, C          ; plot 2D
ADD 1, A
ADD 10, Z             ; move into Z plane
UEQUAL 768, A   ; … quite a ways
ADD 20, X             ; next wave along
ADD 1, C              ; is a different colour
ADD 5, D              ; modulate subsequent waves
IFEQUAL 16, C
SET 1, C
ENDIF                 ; avoid black
WAIT 1                ; so you can break in
UGREATER_THAN 400, X
```

PSTATE: Return both Symmetry States
Syntax: PSTATE <var1>, <var2>

PSTATE returns, in the two specified variables, the current states of the pre-sym <var1> and post-sym <var2> sections of the main plot routine. The information is returned as binary bit patterns.

Example:

PSTATE A, B

Might return]00000011 in A as the pre-sym state, and]11111111 in B as the post-sym state.

RETURN: Return from a subroutine
Syntax: RETURN

The RETURN command, when encountered, returns execution to the routine which called the subroutine. If the RETURN command is encountered when no subroutine has in fact been called, it is ignored.

Associated command: ROUTINE

RND: Return a random number in a specified range
Syntax: RND <range>, <variable>

The RND command allows you to generate a random number in a specified range. The range is assumed to straddle the zero-point, and so if 256 is specified as the range, random numbers will be returned in the range +/- 128.

Examples:

RND 10, A

Would leave A containing +/- 5

RND 100, A
ADD 50, A

Would generate a random number 0-100

ROTATE_PALETTE: Rotate a section of the current palette
Syntax: ROTATE_PALETTE <start-col>, <end-col>, <direction>

This command will rotate a specified portion of the current palette to the left or right according to the value of the DIRECTION parameter. Mostly you will use the Colour Cooker to perform palette dynamics, but this command can sometimes be useful. The parameters <start col> and <end col> are the colour numbers of the colours at each end of the range, the <direction> parameter specifies which way to rotate (negative = rotate to the left, positive = rotate to the right). The absolute value of the direction parameter specifies how far to rotate the palette.

Examples:

ROTATE_PALETTE 1, 15, -1

Rotates the entire palette, apart from colour 0, one colour to the left

```
ROTATE_PALETTE 1, 15, 4
```

Rotates the palette four colours to the right

```
LOOP
ROTATE_PALETTE 1, 7, -1
ROTATE_PALETTE 8, 15, 2
WAIT 1
FOREVER
```

Rotates the palette halves in different directions at different speeds.

Associated commands: STROBE, PALETTE

ROUTINE: Call a KML subroutine
Syntax: ROUTINE <KML prog #>

The ROUTINE command is the equvalent of Basic's GOSUB or machine code's JSR. It transfers program execution to the specified KML program, and upon reaching a RETURN statement, resumes execution at the next statement after the ROUTINE call in the originating program. If no RETURN is found, then execution stops. Subroutines can be nested up to 16 deep.

Example:

Suppose KML slot 20 contained the following:

```
RND 320, X
RND 200, Y
ADD 160, X
ADD 100, Y
RETURN
```

Then the code fragment

```
ROUTINE 20
PLOT X, Y, 8
```

Would plot a dot in a random position on the screen.

Associated commands: RETURN

RUN_SEQUENCER: Start the sequencer at the current point
Syntax: RUN_SEQUENCER

Starts the internal sequencer at its current position. Once running, the Sequencer will execute the KML command sequence recorded in its 8 tracks automatically. Provided tha no state of Interrupt Overrun exists, the Sequencer should be accurate to 1/50 sec.

Example:

```
LOOP
SET BUTTONS, A
AND 1, A
IFNOT_EQUAL 0, A
RUN_SEQUENCER
KILLALL
WAIT 1
ENDIF
WAIT                                                                                    1
FOREVER
```

Waits for the right-hand mouse button to be pressed, and then starts the sequencer.

Associated commands: SEQTIME, STOP_SEQUENCER

SCALE: Re-scale a value in a range
Syntax: SCALE <value>, <range1>, <range2>, <variable>

This command is used to re-scale <value>, When you use this command, <valu> should be in the range 0 - <range1>. The value will be re-scaled to fit in the range 0 - <range2>, and the result placed in the destination <variable>.

Example:

```
FXENABLE 0
CLS
SET 0, V
LOOP
PLOT V, 50, 7
```

```
SCALE V, 320, 200, X
PLOT X, 150, 15
ADD 1, V
UEQUAL 320, V
```

This demonstrates the use of SCALE by stepping through the points on a line of length 320 pixels, plotting them, and simultaneously using the SCALE command to draw a scaled-down line 200 pixels long.

SCOPE: Turn ON and OFF SillyScope traces
Syntax: SCOPE <trace 1>, <trace 2>

The SCOPE command is used to activate and de-activate either or both of the two SillyScope traces. The parameters <trace 1> and <trace 2> determine whether to turn the traces on or off; 0 = off, 1 = on. Generally, when using the SillyScope, you should remember the following points:

Laser-mode SillyScope traces use the system's Line-A linedraw routines instead of my own plot routines; the system routines are faster, but they don't take into account any foreground and background graphics you may have displayed with the PICTURE command. Nor do they respond to any section of the Main Plot Routine, so you can't put them on REMAPs or CUBEs. Unfortunate, but that's the price you have to pay for the extra speed.

Laser-mode traces take their symmetry settings from the settings of the Pre-Symmetry module of the Main Plot Routine, regardless of whether the Pre-Sym is activated or not. This allows you to have different settings of symmetry for Laser-mode and Dot-mode traces (Dot-mode stuff DOES respond to the full Main Plot Routine).

Both types of SillyScope traces are 'unclean' in that if you change symmetry they are apt to leave pixels around on the display. To avoid this, you should use

```
SCOPE 0, 0
WAIT 1
SYM whatever
SCOPE 1, 1
```

Which ensures that the traces are turned off when the symmetry is changed.

Oh, yeah, one more point, you can use the SCOPE command to change the state of only one trace; if you specify –1 as either parameter then the current setting of that trace is left unmodified. For example

```
SCOPE 0, -1
```

Would turn off trace #1. whilst doing nothing at all to trace #2.

Associated commands: XWAVE, YWAVE, XSWEEP, YSWEEP, XPHASE, YPHASE, XAMPLITUDE, YAMPLITUDE, XCENTRE, YCENTRE, SCTRAIL, COLOUR

SCTRAIL: Set SillyScope trailback oscillator or constant
Syntax: SCTRAIL <trace>, <-1>, <const>, <0>, <0> **for constant**
Syntax: SCTRAIL <trace>, <wave #>, <start>, <speed>, <amplitude> **for oscillator**
See section on XSWEEP for explanation of syntax and full example

SEND: Tell the system to send its mouse data out on MIDI
Syntax: SEND

When the SEND command is executed, the system will begin sending out, over MIDI, data packets containing the mouse position and button states, tagged with the sending machine's node number. The remote machine can then be LINKed to the mouse movements of the sending machine.

Associated commands: UNSEND, LINK

SEQTIME: Set the sequencer to a certain point in time
Syntax: SEQTIME <m-hi>, <m-lo>, <s-hi>, <s-lo>, <f-hi>, <f-lo>

SEQTIME allows you to set the current sequencer time to wherever you want. The six parameters following the SEQTIME commands are just the minutes, seconds and 1/50 seconds of the time expressed as single digits.

Examples:

```
SEQTIME 0, 0, 0, 0, 0, 0
```

Sets the sequencer back to start

```
SEQTIME 2,3, 0, 4, 4, 2
```

Sets the sequencer to 23 minutes, 4 seconds, 42/50 second.

Associated commands:
RUN_SEQUENCER, STOP_SEQUENCER

SET: Assign a value to a variable

Syntax: SET <value>, <variable>

The SET command assigns the <value> parameter to the variable specified in the second parameter. The variable can be of type alpha, global or system; the value can be in any form.

Examples:

```
SET 0, A            ; Set alpha variable A to zero
SET CX, X           ; Set alpha vairable X to the current x cursor position
SET C, SVX          ; Set system variable SVX (starfield velocity X)
                    ; to the value currently in alpha variable C
```

SINE: Return sine value scaled to a specific value
Syntax: SINE <angle>, <scalar>, <variable>

The SINE command isn’t like (say) SIN in Basic, which returns a floating-point value. Because we are working with integers, SINE behaves differently. The <scale> value specifies the peak-to-peak amplitude of the wave; the wave value at <angle> is calculated and returned as the contents of <variable>. (The angle is specified by values of 0-255, rather than 0-360 as in ‘conventional’ maths).

That sounds weird, but actually it’s easy to understand. Consider the following:

Example:

(Enter this into an empty slot, assign it and run it)

```
CLS
SET 1, C
LOOP
SET 0, V
SET C, D
MULTIPLY 12, D  ; D will be the amplitude of the wave
LOOP
SINE V, D, Y    ; Returns a signed value in Y
SET V, X
ADD 32, X       ; Centre in X
ADD 100, Y      ; Centry in Y
PLOT X, Y, C
ADD 1, V
UEQUAL 256, V   ; Repeat for all angles
ADD 1, C
WAIT 1
UEQUAL 16, C    ; Do fifteen waves of different colours
```

Associated commands: COSINE

SLICE: Draw section of VSEQ frame
Syntax: SLICE <dest>, <v-pos>, <source>, <v-start>, <height>

This flexible command is most useful, allowing you to scroll smoothly through frames of VSEQ, or move sections of pictures under mouse control. The parameters are set upas follows: <dest> is the frame-number of the VSEQ frame to place the resultant image in; <v-pos> is the position on the destination screen at which you want the image to start being drawn; <source> is the frame # of the image you’re taking; <v-start> is the line from which you’ll start extracting the image data; and <height> is the number of lines of image data, from the <v-start> position, you want to transfer. If the settings of <v-start> and <height> mean you run off the bottom of the source frame, data is taken from the top of the successive frame.

Examples:

A program to scroll smoothly through frames 0-7 of VSEQ. Uses frame #10 as the buffer to build the image in.

```
LOCK                ; in case frame 10 has no palette, keep current
DISPLAY_FRAME 10    ; show the buffer frame
SET 0, F
LOOP
SET 0, L
SLICE 10, 0, F, L, 200
WAIT 1
ADD 1, L
UEQUAL 200, L
ADD 1, F
UEQUAL 8, F
```

Now, a modified version of the same program, adding double-buffering for a flicker-free display, and overlaid on the scrolling, a ‘window’ into the top half of frame 0, which can be moved around by the mouse:

```
LOCK
SET 1, D             ; D is a flag used to flick between buffer screens
SET 0, F
LOOP
SET 0, L
LOOP
SET 10, B
ADD D, B             ; find correct buffer
SLICE B, 0, F, L, 200     ; the scroll
SLICE B, CY, 0, 0, 100; the window
DISPLAY_FRAME B      ; display the buffer
ADD 1, D
AND 1, D             ; flip the buffer pointer
WAIT 1
UEQUAL 200, L
ADD 1, F
UEQUAL 8, F
```

SLICE is a very powerful command. Enjoy.

SMART_BOMB_VSEQ: Clear and reallocate all available VSEQ memory
Syntax: SMART_BOMB_VSEQ

Smartbombing the VSEQ will de-allocate all .MAPs, .BAA files, and restore the VSEQ to maximum capacity. The picture frames are not actually cleared, although they are listed as de-allocated, so it’s possible to recover picture info even after an erroneous SMART_BOMB_VSEQ command.

SNAPSHOT: Copy the contents of the screen to VSEQ frame
Syntax: SNAPSHOT <VSEQ Frame #>, <merge #>

Upon execution, this command copies the contents of the screen to the specified VSEQ frame, according to the setting of the MERGE flag. MERGE = 0 means ‘replace the frame with the screen contents’, MERGE = 1 means ‘merge the screen contents with the existing contents of the specified frame’.

Examples:

SNAPSHOT 0, 0

Transfer the screen contents to VSEQ frame 0

SNAPSHOT 3, 1

Merge the screen contents with the contents of frame #3

SPIN: Rotate co-ordinates around a given point
Syntax: SPIN <x-co>, <y-co>, <x-cen>, <y-cen>, <angle>, <d-x>, <d-y>

Yow! Plenty parameters! This command allows you to take a pair of co-ordinates, and SPIN them around a specific point by a given angle; then place the results in another co-ordinate pair. The parameters <x-co> and <y-co> are the starting point, <x-cen> and <y-cen> specify the centre of rotation; <angle> is the amount, in Yakly degrees, to spin the co-ordinates by; and <d-x> and <d-y> are variables to receive the result.

Example:

(SPINs a triangle around the current cursor position)

```
CLS
SET 0, A
SET 1, C
LOOP
SET CX, X
SET CY, Y
SET CX, I
SET CY, J
SUBTRACT 20, X
SUBTRACT 20, Y
ADD 20, I
ADD 20, J
SPIN X, Y, CX, CY, A, X, Y
SPIN I, J, CX, CY, A, X, Y
```

```
TRIANGLE X, Y, I, J, CX, CY, C
ADD 5, A
ADD 1, C
IFEQUAL 16, C
SET 1, C
ENDIF
WAIT 1
FOREVER
```

SSTATE: Return current SillyScope state
Syntax: SSTATE <var1>, <var2>

This command returns the current state of the SillyScope traces in the specified variables. The results are zero if a trace is off, one if a trace is on.

Example:

```
SSTATE A, B
```

Leaves trace 1 state in A, trace 2 state in B

STARFIELD: Activate or shutdown starfield generation
Syntax: STARFIELD <switch>

The STARFIELD command is used to turn on a starfield you may have generated previously using either the Starfield Editor or the Starfield system variables. If the <switch> parameter is 1, then the starfield is activated. If <switch> is 0, then the starfield is shut down. If <switch> is –1, then a 'hard' starfield shutdown is performed.

The reason for the two different shutdown commands is because a starfield doesn't turn off instantly. It can take from one to eight frametimes to disappear, depending on the starfield's Trailback setting. The 'safe' way to shutdown a starfield is

```
STARFIELD 0
WAIT 8
```

However, you can 'force' a starfield to shutdown instantly by

```
STARFIELD –1
```

But it will leave star images on the screen, so you may wish to follow it with a CLS command.

Associated system variables:

SVX, SVY, SVZ, ANG_VELOCITY, ROX, ROY, TRAIL, SCLOW, SCHIGH, SCMODE, SCRATE, INX, INY, INZ, ABS_ANGLE, INRATE, ZCLIP, PROJECT_DEPTH

Associated commands: ISTAR

STOP_SEQUENCER: Stop the internal sequencer
Syntax: STOP_SEQUENCER

This command disengages the internal sequencer, it stps executing KML events.

Associated commands: RUN_SEQUENCER, SEQTIME

STROBE: Set colour within current palette
Syntax: STROBE <colour #>, <red>, <green>, <blue>

The STROBE command is used to set any of the 16 colours in the current palette. The first parameter is the logical colour number, 0-15; the following three values give the red, green and blue values respectively.

Examples:

```
STROBE 0, 0, 0, 0         ; Set background to black
SET 7, R
SET 0, G
SET 7, B
SET 1, C
LOOP
STROBE C, R, G, B
ADD 1, C
UEQUAL 8, C       ; Set colours 1-7 to purple
```

Associated commands: CGET, PALETTE, ROTATE_PALETTE

SUBTRACT: Subtract a value from a variable
Syntax: SUBTRACT <value>, <variable>

This command is for subtracting the specified value from a destination variable. The value can be in any formatl the variable can be alpha, sys or global.

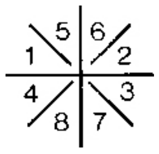
Examples:

SUBTRACT 1, V ; Subtract 1 from variable v

SUBTRACT CX, X ; Subtract system variable CX from alpha X

SYM: Set main plot routine symmetry
Syntax: SYM <sym mode>, <bit pattern>

The SYM command is used to set the symmetry for the main plot routine. The parameter <sym mode> is zero for setting pre-symmetry, and 1 for post-symmetry. The <bit-pattern> is an eight-bit binary value, with each of the eight bits corresponding to the following symmetry nodes:



Examples:

SYM 1,]00001111 ; Set post-symmetry to quad-sym

SYM 0,]10100101 ; Set pre-symmetry to spiral sym

Associated commands: FXENABLE

SYNTH: Set MIDI to listen to external synthesiser
Syntax: SYNTH <switch>

The SYNTH command should be issued to prepare the system to receive MIDI note data from an external synthesiser. The <switch> parameter switches the SYNTH mode on or off; a 1 means ‘set synth mode’, 0 means ‘set Trip-A-Tron networking mode’.

Associated commands: MCHAN, MPITCH, MVELOCITY, MBEND, POLY

TRIANGLE: Draw a filled triangle on the current screen
Syntax: TRIANGLE <x1>, <y1>, <x2>, <y2>, <x3>, <y3>, <colour>

Draws a filled triangle, with corners at the co-ordinates <x1>, <y1>, -<x2>, <y2>, -<x3>, <y3>, in ST screen colour <colour>. Uses the line-A for the draw operation; so the main plot routine settings, including symmetry, do no affect it; nor will it respect background/foreground PICTUREs.

Here’s a wee demo-ette for you:

```
CLS
SET 200, R
SET 0, A
SET 1, C
LOOP
SINE A, R, X
COSINE A, R, Y
ADD 160, X
ADD 100, Y
SET 320, W
SUBTRACT X, W
TRIANGLE X, Y, W, Y, 160, 100, C
ADD 4, A
ADD 1, C
SUBTRACT 1, R
UEQUAL 0, R
```

Very pretty with a dynamic colour flow, that one.

TWEEN: Calculate intermediate variables in a range
Syntax: TWEEN <s-val>, <e-val>, <# steps>, <c-step>, <variable>

The TWEEN command allows you to determine intermediate points between two end values. The end values are passed in <s-val> and <e-val> and the total number of steps along the range is passed in <# steps>. The current step number is passed in <c-step>, and the resultant value is placed in <variable>.

Example:

Draw 50 evenly-spaced points along the line (23, 10) – (230, 190).

```
FXENABLE 0
SET 0, V
LOOP
TWEEN 23, 230, 50, V, X
TWEEN 10, 190, 50, V, Y
PLOT X, Y, 8
ADD 1, V
UEQUAL 50, V
```

UEQUAL: Loop closure, Until-Equal
Syntax: UEQUAL <test value 1>, <test value 2>

The UEQUAL statement compares the two test values, and, if they are **not** equal, transfers program execution back to the start of the LOOP.

Example:

```
SET 0, V
LOOP
…
…
ADD 1, V
UEQUAL 8, V       ; Executes the code in the loop eight times
                  ; And dropping out when V = 8
```

Associated commands: LOOP

UGREATER_THAN: Loop closure, UNTIL GREATER THAN
Syntax: UGREATER_THAN <test value 1>, <test value 2>

The UGREATER_THAN command compares the two test values, and if value 2 is greater than value 1, passes control to the next statement; any other case results in the LOOP continuing.

Example:

```
SET 0, V
LOOP
…
… KML CODE
…
WAIT 1            ; Executes the code in the loop
ADD 1, V          ; Repeatedly, until V > 10
UGREATER_THAN 10, V
```

Associated statements: LOOP

ULESS_THAN: Loop closure, UNTIL LESS THAN
Syntax: ULESS_THAN <test value 1>, <test value 2>

The ULESS_THAN command compares the two test values and, if value #1 is greater than or equal to value #2, executes the LOOP again. If value #1 is less than value #2, then execution passes to the next statement after the ULESS_THAN command.

Example:

```
LOOP
…
…                 ; Executes the code inside the loop
WAIT 1            ; until the cursor goes below the line
ULESS_THAN 30, CX   ; x = 30
```

Associated command: LOOP

UNEQUAL: Until Not Equal loop closure
Syntax: UNEQUAL <value 1>, <value 2>

Compares <value 1> and <value 2>; if they are not equal, LOOPs back for another iteration, if they are not equal, passes command on to the next statement.

Example:

```
LOOP
SET BUTTONS, A
AND 3, A
WAIT 1
UNEQUAL 0, A
```

Waits until either Mouse Button is present.

Associated command: LOOP

UNLOCK: Release previously locked palette
Syntax: UNLOCK

UNLOCK is the complement of the LOCK command. It is used to free the main palette setting after it has been LOCKed. See the description of the LOCK command for more details on locking palettes.

Associated commands: LOCK

UNPACK: Decompact a compressed picture from RLE-stash
Syntax: UNPACK <stash-frame #>, <VSEQ frame #>, <merge>

This command is used in conjunction with RLE-stashes (.BAA files). The first parameter specifies the number of the Stashed frame to de-compact; the second parameter specifies which frame of VSEQ to place the de-compacted image in. The third <merge> parameter will, if set non-zero, merge the decompacted image with any image already occupying the destination VSEQ frame.

Examples:

```
UNPACK 2, 0, 0
DISPLAY_FRAME 0
```

Unpacks and displays frame 2 of an RLE-stashed sequence

```
DISPLAY_FRAME 0
LOOP
SET 0, A
LOOP
UNPACK A, 0, 0
WAIT 2
ADD 1, A
UEQUAL 20, A
FOREVER
```

Continually displays a 20-frame RLEstashed sequence.

Associated commands: PICTURE, DISPLAY_FRAME

UNSEND: Cease sending mouse data on MIDI
Syntax: UNSEND

Upon execution of the UNSEND command, the system will cease to send mouse position data packets out over MIDI. Before you UNSEND, you should ensure that all slave systems have unlinked (see LINK).

Associated commands: SEND, LINK

USER_WAVE: Return value from user-defined waveform table
Syntax: USER_WAVE <wave-#>, <index>, <variable>

This function allows you to treat any of the eight waves in the waveform editor as functions, each wave returning a value of +/- 127. The three parameters are as follows: <wave-#>, 1-8, represents the waveform number 1; <index> is an index into the 256 entries comprising each wave (use values 0-255); and <variable> is the variable in which to place the returned value.

Example:

```
USER_WAVE 3, 64, A
```

Will return the 64 th value from the table defining waveform 3.

Associated commands: MWAVE

VPEEK: Return the colour of a given pixel
Syntax: VPEEK <x-co>, <y-co>, <variable>

Upon encountering VPEEK, the system examines the colour of the pixel specified by the co-ordinates <x-co> and <y-co>. The colour is returned in the contents of <variable>

Example (just to prove it works)

```
FXENABLE 0
PLOT 100, 100, 6
```

```
VPEEK 100, 100, A
NUMERIC 10, 10, A
```

Will print up '6'

WAIT: Releas CPU for specified interval
Syntax: WAIT <time in 1/50 sec>

The WAIT command is used to break up your KML code into 'time-slices'. Upon reaching a WAIT statement, execution of the KML program ceases for the specified duration, freeing up the system to carry on running other KML progrs and Trip-A-Tron effects.

Examples:

```
WAIT 1              ; Releases CPU for one frametime
SET 7, F
LOOP
STROBE 0, F, 0 0
WAIT 3
SUBTRACT 1, F
ULESS_THAN 0, F     ; Performs a 'soft' strobe
```

WAKE_UP: Wake up remote networked Trip-A-Trons
Syntax: WAKE_UP

The WAKE_UP command will cause any MIDI-linked external Ataris which are still in the 'PRESS LEFT MOUSE BUTTON FOR MOUSE' display stage to load their DEFAULT.EWE macros, load the files, drop their panels and attempt to execute their KML #127 programs.

XAMPLITUDE: Set SillyScope X amplitude wave for constant
Syntax: XAMPLITUDE <trace>, <-1>, <const>, <0>, <0> **for constant**
Syntax: XAMPLITUDE <trace>, <wave>, <start>, <speed>, <scale> **for an oscillator**
See the XSWEEP section for explanation of syntax and full demo.

XCENTRE: Set SillyScope trace to X centre
Syntax: XCENTRE <trace>, <-1>, <const>, <0>, <0> **for constant**
Syntax: XCENTRE <trace>, <wave>, <start>, <speed>, <scale> **for an oscillator**
See the XSWEEP section for explanation of syntax and full demo.

XOR: Perform exclusive-OR
Syntax: XOR <source>, <dest-var>

XOR performs an exclusive-OR of <source> on the contents of <dest-var>, leaving the result therein.

Example:

```
SET ]00001111, A
XOR ]01010101, A
```

Would leave A holding the value [01010000

Associated commands: AND, OR

XPHASE: Set SillyScope X phase wave or constant
Syntax: XWAVE <trace>, <-1>, <const>, <0>, <0> **for constant**
Syntax: XWAVE <trace>, <wave>, <start>, <speed>, <scale> **for an oscillator**
See the XSWEEP section for explanation of syntax and full demo.

XREFLECT: Reflect the X-co-ordinate given in the Y axis
Syntax: XREFLECT <co-ord>

This is really a convenience command, saving you a couple of lines and a variable. You pass it a co-ordinate, and it returns the co-ordinate reflected in the Y-axis.

Example:

Where you might have the code segment

```
SET 319, R
SUB X, R
SET R, X
```

To generate the reflection of X, you can put

```
XREFLECT X
```

To get the same result.

Associated commands: YREFLECT

XSWEEP: Set SillyScope X sweep wave or constant
Syntax: XSWEEP <trace-#>, <-1>, <constant>, <0>, <0> **for a constant**
Syntax: XSWEEP <trace-#>, <wave-x>, <start>, <speed>, <scale> **for a waveform**

SETTING SILLYSCOPE OSCILLATORS: General Notes
The commands XSWEEP, YSWEEP, XPHASE, YPHASE, XAMPLITUDE, YAMPLITUDE, XCENTRE, YCENTRE, SCTRAIL and COLOUR all have similar, somewhat complex syntax, which I shall explain here.

Each SillyScope oscillator can be set to be just that: an oscillator, running from a specified waveform, at a certain speed and producing results of a particular amplitude; or it can be a constant. The two different syntaxes of these SillyScope commands reflect these to modes.

Setting a Constant: the <trace-#> parameter specifies the trace number on which you're setting the parameter; 1 for trace 1, 2 for trace 2. The next parameter must be –1, which tells the system set a constant, not a waveform. The third parameter should be the constant itself. The last two parameters can be anything; they are not used in setting up a constant.

Thus.

```
XPHASE 2, -1, 64, 0, 0
```

Would set the XPHASE of SillyScope trace 2 to a constant 64.

Setting an Oscillator: all the parameters are used <trace-#> specifies 'scope trace number 1 or 2, as before; <wave #> specifies the system wave number of the waveform to use for the oscillator (1-8), <start> is a phase-offset into this waveform, set at the time the command is executed; you usually leave it as zerol <speed> is the step-value through the 256 entries that make up a waveform data table, -1 would look at every value, 8 every eighth value, and so forth; finally <ampliude> specifies **half** of the maximum peak-to-peak amplitude of the wave (i.e an amplitude setting of 7 would produce a wave 14 units high).

Thus:

```
XPHASE 1, 4, 0, 1, 50
```

Would start an oscillator for the X phase of trace 1 using waveform 4, at speed 1, with an amplitude 50 (100 peak-to-peak).

Generally you will get your SillyScope settings from the SillyScope screen by noting down the parameters and waveform numbers for each function once you've used the edit controls to set up a good trace. A comple SillyScope setup might look like this:

```
SCOPE 0, -1
WAIT 1                   ; Ensure old trace removed before changing stuff
SYM 1, ]00000001         ; Set pre-sym for laser mode
LASER 1, 1
XWAVE 1, 1
YWAVE 1, 1
XSWEEP 1, -1, 12, 0,0    ; set X sweep to constant 12
YSWEEP 1, -1, 16, 0,0    ; set Y sweep to constant 16
XPHASE 1, 1, 0, 1, 128
YPHASE 1, 1, 0, 1, 128   ; X and Y phase oscillators
XAMPLITUDE 1, 3, 0, 1, 60
YAMPLITUDE 1, 2, 0, 1, 100      ; Amplutude oscillators
XCENTRE 1, 1, , 1, 109
YCENTRE 1, 1, 0, 1, 59   ; Centreing oscillators
SCTRAIL 1, -1, 2, 0, 0   ; Trailback const 2
COLOUR 1, 5, 0, 15, 7    ; Colour oscillator
SCOPE 1, -1
```

This code is a complete SillyScope laser mode trace setup; it uses the DEMO.WAV waveform set if you want to try it out!

Commands associated with SillyScope:

SCOPE, XWAVE, YWAVE, XSWEEP, YSWEEP, XPHASE, YPHASE, XAMPLITUDE, YAMPLITUDE, XCENTRE, YCENTRE, SCTRAIL, COLOUR

XWAVE: SillyScope base X wave setting
Syntax: XWAVE <trace-#>, <waveform-#>

XWAVE sets the base waveform in X for either of the two SillyScope traces. The <trace-#> parameter determines which of the two traces to apply the waveform to (1 or 2), and the <waveform #> number corresponds to one of the system waveforms, 1-8.

Example:

```
XWAVE 1, 3
```

Use Waveform #3 as the base X waveform in SillyScope trace #1.

See also the complete SillyScope setup demo in the XSWEEP section.

Associated commands: YWAVE, XSWEEP, YSWEEP, XPHASE, YPHASE, XAMPLITUDE, YAMPLITUDE, XCENTRE, YCENTRE, SCTRAIL, COLOUR

YAMPLITUDE: Set SillyScope Y amplitude wave for constant
Syntax: YAMPLITUDE <trace>, <-1>, <const>, <0>, <0> **for constant**
Syntax: YAMPLITUDE <trace>, <wave>, <start>, <speed>, <scale> **for an oscillator**
See the XSWEEP section for explanation of syntax and full demo.

YCENTRE: Set SillyScope trace Y centring wave for constant
Syntax: YCENTRE <trace>, <-1>, <const>, <0>, <0> **for constant**
Syntax: YCENTRE <trace>, <wave>, <start>, <speed>, <scale> **for an oscillator**
See the XSWEEP section for explanation of syntax and full demo.

YPHASE: Set SillyScope Y phase wave for constant
Syntax: YPHASE <trace>, <-1>, <const>, <0>, <0> **for constant**
Syntax: YPHASE <trace>, <wave>, <start>, <speed>, <scale> **for an oscillator**
See the XSWEEP section for explanation of syntax and full demo.

YREFLECT: Reflect the Y-co-ordinate given in the X axis
Syntax: YREFLECT <co-ord>

This command, the partner of XREFLEC is a timesaver designed to save you a couple of lines of code and a variable. You pass it a Y co-ordinate, and it returns the reflection of that co-ordinate in the Y-axis.

Example:

Where you could code

```
SET 199, R
SUB Y, R
SET R, Y
```

To generate the reflection of Y, you can put

```
YREFLECT Y
```

To get the same result.

Associated commands: XREFLECT

YSWEEP: Set SillyScope Y sweep oscillator or constant setting
Syntax: YSWEEP <trace>, <-1>, <const>, <0>, <0> **for constant**
Syntax: YSWEEP <trace>, <wave>, <start>, <speed>, <amplitude> **for an oscillator**
See the XSWEEP section for explanation of syntax and full demo.

YWAVE: SillyScope base Y wave setting
Syntax: YWAVE <trace-#>, <waveform-#>

YWAVE is used to desiginate the base Y waveform on either SillyScope trace. The <trace-#> parameter should be 1 for trace 1, 2 for trace 2; the <waveform-#> parameter corresponds to one of the system waveforms, 1-8.

Example:

```
YWAVE 1, 2
```

Use Waveform number 2 as the base Y waveform in SillyScope trace #1.

See the XSWEEP section for a complete SillyScope setup demo

Associated commands: SCOPE, XWAVE, XSWEEP, YSWEEP, XPHASE, YPHASE, XAMPLITUDE, YAMPLITUDE, XCENTRE, YCENTRE, SCTRAIL, COLOUR

THE KML SYSTEM VARIABLES

There now follows a list of the system variables recognised by KML, along with an explanation of the function of each.

CX: Cursor X position
The current X position of the mouse cursor is contained in this variable.

CY: Cursor Y position
The current Y position of the mouse cursor is contained in this variable.

BUTTONS: Mouse Button States
This variable contains the current status of both the Mouse Buttons. Only the lowest 2 bits are significant; bit 1 provides the right-hand button state (1 = pressed); bit 2 is the left-button state. You can check for individual button presses by using the AND function:

```
SET BUTTONS, A
AND 1, A
```

Would leave A non-zero if the right Mouse Button were pressed.

```
SET BUTTONS, A
```

AND 2, A

Would leave A non-zero if the left Mouse Button were pressed.

SVX: Star Velocity (X) (Legal +/- 500)
This variable contains the current X-speed of the stars in any displayed starfield.

SVY: Star Velocity (Y) (Legal _+/- 500)
Vertical (Y) speed of starfield.

SVZ: Star Velocity (Z) (Legal +/- 500)
Z-speed of stars (into or out of the screen)

ANG_VELOCITY: Angular Velocity of stars (Legal +/-500)
Rotational velocity of starfield.

ZCLIP: Starfield Z-Clipping Plane (Legal 0-10000)
The position of the furthest allowable position away in the Z-plane, after which stars disappear.

ROX: Rotation Point (X) of Starfield (Legal +/- 500)
Point about which the starfield rotates.

ROY: Rotation Point (Y) of Starfield (Legal +/- 500)
Point about which the starfield rotates.

PROJECTION_DEPTH: Starfield Projection Depth (Legal –1 to –10000)
Controls the intensity of the perspective effect for starfield 3.

ABS_ANGLE: Absolute starfield rotate angle (Legal 0-255)
The angle at which the starfield is displayed.

INX: Star Init Point (X) (Legal +/- 320)
Point of star origin. –1 = random origin.

INY: Star Init Point (Y) (Legal +/- 200)
Point of star origin. –1 = random origin.

INZ: Star Init Point (Z) (Legal –1 to 10000)
Point of star origin. –1 = random origin.

INRATE: Star Init Rate (Legal 0-1000)
Frequency of star initialisation (the lower, the more frequent, except for zero (or –1, which is illegal, but ain’t all the best things in life?) in which case ISTAR must be used).

SCLOW: Star Colour (Low) (Legal 0-15)
Star colour of generated stars.

SCHIGH: Star Colour (High) (Legal 0-15)
Maximum colour of stars.

SCMODE: Mode of starfield colour transition (Legal 0-1)
If zero, star colour ‘wraps’ on reaching maximum; if one, star colour ‘bounces’.

SCRATE: Rate of star colour transition (Legal 1-1000)
Rate at which stars transit thru colours. The lower the faster.

TRAIL: Star trailback length (Legal 0-7)
Length of pixel trail behind each star.

For a more detailed explanation of the starfield variables, see the section on the Starfield Edit screen. You’ll find plenty of examples of setting up and manipulating ‘fields from KML if you look at the KML progs in the starfield demo.

PATTERN: Current decay, line or expandor number (Legal 1-64)
Specifies which shape to use for pattern generation.

General Notes on using Trip-A-Tron

1: The MIDI Factor

Trip-A-Tron can be used with MIDI in two different modes; SYNTH mode and Networking mode. The SYNTH mode allows you to listen to external MIDI events over sixteen MIDI channels, and take action via KML according to what is going on, allowing you to moderate effects according to what is going on on the synth keyboard, or just use a dummy MIDI keyboard as an ‘effects bank’. The current MIDI handling is very basic, and I may well extend it in future updates of the program, but there’s enough there to allow some experimentation already. You connect your synth to Trip-A-Tron by just plugging it into the MIDI IN port on your ST, and using either the Midi Test screen or KML to set the system into SYNTH mode. The KML commands to read pitch and velocity events are detailed in the KML command summary.

In Networking mode, you can link up to eight STs together and control them all from a single ‘master’ machine. This really comes into its own if you’re getting really serious about using Trip-A-Tron and want to invest in a video mixer like mine (it ain’t cheap, but it is EXCEEDINGLY groovy – see notes below). To link the multiple machines, you ‘daisy chain’ them from the master machine: connect the MIDI OUT of the master into the MIDI IN of the first slave; them the MIDI OUT of slave 1 to MIDI IN of slave 2, and so on up to the limit of your machines. Assign each machine a different node number using the Midi Test screen or the KML command MCHAN; then you can use the EXTERN and LINK commands to controlany or all the machines from the one master machine.

2: Using the PRO-DRAW graphics tablet

If you have this graphics tablet, connect it up and boot TRIP-A-TRON. At the intro screen, click the **right** Mouse Button to put the system in tablet mode. You’ll find the tablet the perfect controller for the effects; much smoother and more precise control of pattern generation is possible than with a mouse. (In using the tablet, the left mouse button is simulated by the button on the barrel of the stylus; the nib sensor acts as the right mouse button). An advantage of using the tablet is that, as there is a one-to-one correlation between the pen’s position on the tablet and the cursor’s position on the screen, you never get ‘lost’ and thus you can switch off the visible cursors leaving just pure unadulterated pattern.

3: The video mixer

If you get serious about using Trip-A-Tron commercially, you might be interested in this excellent piece of hardware which has been designed for used with Trip-A-Tron. The mixer allows you to connect up to five Atari STs together and display their outputs simultaneously on one display. Thus, with five Ataris running through the mixer, you have the equivalent of the ‘super-machine’ with 5 68000s and a display supporting 80 simultaneous colours! The mixer can fade any of the five channels up or down, and provides R, G, B, Sync outputs, plus a separate monitor channel allowing you to preview any channel before you feed it into the mix. We’re just now looking at building in a high-quality composite-video output (for videoing the display) and a Genlock. Belive me, if you like Trip-A-Tron on a single machine, wait’ll you try it on multiple machines through the mixer… Anyone serious about using Trip-A-Tron in this way should get in touch for more info and maybe a demo.

Other Demos on your Disks

There may well be other demos in your program disks; I shall probably include more and better demos as I develop them. To check out the demos on your particular disks, look on the disks from the Disk Screen and check for extra .EWE files apart from the standard DEFAULT.EWE and DEMO2.EWE files. Any that you see, LOAD in and then do LOAD MACRO. Return to the Control Panel, and bring up the Key Assign panel to see which keys to press to see the demos. (To get you started, look at the KMLDEMO.EWE demo set; this contains many of the examples listed in the KML Command Summary. The RETURN key should ne used between demos when running KMLDEMO.EWE: it does a KILLALL, stopping all parallel channels).

Summary of SaveStuff

When you want to save your work, here are the extenders to use in the Disk screen, laid out in a quick-reference table:

.NEO	NeoChrome pictures
.PI1	DEGAS low-resolution pictures
.RLE	Run-Length Encoded pictures
.KML	KML program sets
.MAP	Screen re-mapping files
.PAT	All pattern data
.PAL	200 palettes and the nine Colour Channels
.YAK	Sequencer data
.BAA	.RLE stashes
.PRE	Global parameter banks and Resize windows
.WAV	The eight System Waveforms
.EWE	Macro Load files

The SHEEP.PRG program

The little SHEEP.PRG program is included on your disks. It is a stand-alone player for RLEstashes (.BAA files). When loaded, it asks for a data disk; place any disk containing a .BAA file in the drive, click the mouse and, memory permitting, it will be loaded and animated. The speed and direction of animation are controllable; full instructions appear on the screen.

Bugs

This is a very complex system, and although I have tried to debug it as fully as I possibly can, it would be hopelessly audacious of me to claim the code is 100% error-free. You could be of immense help to me if, should you come across any nasties, you could let me know, preferably with such details as: what you were doing to get the bug, in which module of the program you were at the time, and suchlike. If you find that a chain of events always produces an error, then please list the events leading up to the glitch. In this manner, I can fix glitches in future releases. Thank you.

Futures

This is the first release of Trip-A-Tron. I see this as a good starting point from which to expand and enhance the system; I'll be using the system a lot over the next months, and I'll doubtless come across things I need, effects I want, and bolt them in. When sufficient new bits are incorporated I'll release an update to the system, and will probably continue to do so for a good time to come. Updates will be available for a nominal fee; we'll let you know when a new version is ready.

Beyond that, who knows? I keep trying to wrap my brain around the concept of an ABAQ with twelve transputers and 16.7 million colours, and the sort of lightsynth I'm gonna write for that particular beastie… belive me, if I don't drop dead first, I'm gonna do it, and when I do, you'll know it fur SURE!

Also in the pipe is the project which, at least in part, led to the creation of Trip-A-Tron: MERAK, an audio-visual experience…

This is a joint project which started over a year ago… shortly after Colourspace was released on the ST, I met with a composer, Adrian Wagner, and we got talking about the possibility of using the lightsynth and his music together to make a combined light/sound album, to be released on video. Through the last year we've worked together, he on the album 'MERAK' and me on Trip-A-Tron. In many ways, this program was designed for the album, and the music was designed for the lightsynth… our concept was ambitious from the start, and I soon came to realise that no mere extension of Colourspace would be sufficient to realise what we wanted to do (originally, I was just going to bolt some extras onto Colourspace and release an update as 'Colourspace 2'). Hence, a clean start, a year's hard work, and at last, Trip-A-Tron.

The music is now complete, so is the lightsynth; we hope to have the video available sometime this August or so. Until then, the music is available nowl if you want some really good lightsynth music (or, indeed, just a great album) you can't do better than this – it was designed for Trip-A-Tron, after all…

You can obtain the album from us, on the following formats:

Compact Disc: £9.95
Cassette: £6.95
Digital Audio Tape (DAT): £14.95

And if you get ito that, you should check out the excellent Inca Gold, too – that's the album I took to Peru and Walkman'ed up on Machu Picchu, well recommended.

Once the video is complerte it will be available on VHS, Beta or 8mm at £12.95.

(see if you can spot the sound-effects of Yak playing ST Star Wars in the album, too….. wear headphones).

Credits

Trip-A-Tron was hand-rolled by YAK over a period of a year.

Concept, Design, Graphics, Programming all by YAK
(It’s not often you see that, in these days of programming-by-committee)

Developed in 68000 machine language on a Mega ST4 and SH204 hard disk drive, using HiSoft Devpac (ecellent assembler); NEOchrome for graphics. Digitised llama courtesy of S.A.M (outrageous bit of hardware!)

YAK would like to thank: Atari (for making it possible – there, a nice fat plug, now how about sponsoring me to the tune of another four Megas to go in my nice video mixer?); the parent-creatures (for genesis, help and support, and handling business hassles); Adrian and all the Wagners (for a superb album, enthusiasm to the max, chicken soup and all those cups of tea – ta Emma); Pink Floyd and Roger Waters (for inspiration – I have stood in the presence of Lord my Rog and head His Holy Word); John Lawrence (for hardware the likes of which I only used to dream about); Denis and Boris (for dead mice); everyone who waited patiently for ‘Colourspace 2 – Coming Soon’ (I think maybe you understand why it took me a while now – and to think I used to get by just shunting a few sprites around on a Commodore screen); PG Tips; the entire casts of East Enders and Neighbours; Paul Atredies; everyone I’ve gotten drunk with down the Fox; all the sheep and goats within a 20-mile radius, Brian Aldiss (for the best three books EVER); everyone who DOESN’T eat sheep; and all those friends of the Yak who helped keep me stable, on the beam, and kept me from spinning off during the heaviest project I’ve ever undertaken!

This program is dedicated to the artist within all of us

-- Yak May 1988

Extra Stuff Not In The Original Manual

Hello! Well, that's that then. That took a lot of typing. No, as you might have guessed, this isn't Yak typing but someone else. If you frequent the Llamasoft website forum you'll maybe know me as Piku. The rest of the world knows me as James Grimwood (except a small collection of people in Scotland who call me Sponge for various random reasons). Who am I? Nobody in particular, I just decided one day to type out the entire Trip-A-Tron manual in the hope it is useful to someone, the original paper versions being more rare than colours on an Atari ST. It took me a month and two days to type out and, if Yak allows it, should be available for other people to read pretty soon (you'll know if it is, since you'll be reading it now!).

I just want to thank Mark@thearchive for his paper version which he posted to me, Yak for the original program (and all its subsequent spawn), and everyone on the Llamasoft forum for sending me that rather nice Gamecube (proof that Nice People still exist and that the universe isn't totally broken).

You could do worse than visiting http://www.llamasoft.co.uk right now! (http://www.piku.org.uk isn't a totally bad place either ;-)

I'm off now to copy this several times in case my PC explodes or Word corrupts the file, or something else that is common place in a Windows PC.

-- Piku/james 2003